2

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DATA COMPRESSION BY USING WAVELET TRANSFORMS AND VECTOR QUANTIZATION

by

Alper Erdemir

June, 1993

Thesis Advisor: Professor Murali Tummala

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School; 6b. OFFICE SYMBOL (if applicable) Code 32	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
DATA COMPRESSION BY USING WAVELET TRANSFORMS AND VECTOR QUANTIZATION

12. PERSONAL AUTHOR(S)
Erdemir, Alper

13a. TYPE OF REPORT Master's Thesis	13b. TIME COVERED FROM 09/92 TO 06/93	14. DATE OF REPORT (Year, Month, Day) June 1993	15. PAGE COUNT 85

16. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the autor and do not reflect the official policy or position of the Department of Defense or the United States Government.

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Wavelet Transforms, Vector Quantization, Speech Coding, Image Coding

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This thesis proposes a new analysis/synthesis procedure for speech and image compression. The algorithm applies the discrete wavelet transform to subject data in order to obtain a set of multiresolution wavelet coefficients. The wavelet coefficients are then encoded by using the generalized Lloyd algorithm. The statistical properties of the wavelet coefficients are utilized to determine the number of resolution levels as well as the codebook size at each resolution level. Coding results show that the new procedure provides a significant improvement in the quality of the reproduced data. The tested data includes speech, image, and transient signals.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Tummala, Murali	22b. TELEPHONE (Include Area Code) (408) 656-2645	22c. OFFICE SYMBOL Code EC/Tu

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Data Compression by Using Wavelet Transforms and Vector Quantization

by

Alper Erdemir
Lieutenant J. G., Turkish Navy
B.S., Turkish Naval Academy, 1987

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1993

Author: [illegible]
Alper Erdemir

Approved by: Murali Tummala
Murali Tummala, Thesis Advisor

Charles W. Therrien
Charles W. Therrien, Second Reader

Michael A. Morgan
Michael A. Morgan, Chairman
Department of Electrical and Computer Engineering

ABSTRACT

This thesis proposes a new analysis/synthesis procedure for speech and image compression. The algorithm applies the discrete wavelet transform to the subject data in order to obtain a set of multiresolution wavelet coefficients. The wavelet coefficients are then encoded by using a multiresolution codebook designed using the generalized Lloyd algorithm. The statistical properties of the wavelet coefficients are utilized to determine the number of resolution levels as well as the codebook size at each resolution level. Coding results show that the new procedure provides a significant improvement in the quality of the reproduced data. The data tested includes speech, image, and transient signals.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

The recent emergence of wavelet transforms has encouraged many researchers to seek new application areas for them [Ref. 10-12]. Since discrete wavelet transforms (DWT) are similar to subband coding systems and subband coders have been used in speech and image compression, wavelets have found immediate application in the waveform coding area [Ref. 10-12]. Data compression consists of converting a data sequence into a stream of relatively low bit rate data for transmission over a digital communication channel or storage in a digital media. The DWT itself does not reduce the total bit rate, but it provides a set of wavelet coefficients using an algorithm with pyramidal architecture. Since wavelet coefficients at different resolution levels exhibit different statistical characteristics, by designing a quantizer for each set, we can reduce the bit rate without significant degradation in quality.

In this thesis, we present a coding algorithm that combines the DWT and vector quantization (VQ). VQ is a method of quantizing a set of data samples jointly as a vector. The overall procedure is known as an *analysis-synthesis method* and is extensively used in transform and subband coding areas. It involves two steps. First, we use the discrete wavelet transform to obtain a set of wavelet coefficients; second, we vector quantize these coefficients by using codebooks designed for each set of coefficients. The application of wavelet transforms to this technique is relatively new. The DWT has been applied to image coding by combining scalar quantization and vector quantization [Ref.

10]. Another algorithm has been reported that uses lattice vector quantization [Ref. 11]. The algorithm we present here uses generalized Lloyd algorithm [Ref. 8] for vector quantization of wavelet coefficients. We apply this coding technique both to speech and to image data.

The remainder of this thesis is organized as follows. In Chapter II, we introduce the basic principles of wavelet theory. This chapter also covers the algorithms for the discrete wavelet transforms in both one and two dimensions. Chapter III describes the basic concepts of vector quantization and presents the vector quantization algorithm used in this thesis. In Chapters IV and V, we combine these algorithms to develop coding methods for speech and image compression. Chapter VI presents conclusions.

II. WAVELET TRANSFORMS

A. INTRODUCTION

Currently wavelet transforms are a popular topic of research in various signal processing applications. They are viewed as a new way to represent signals as well as a new technique for time-frequency analysis. From the multiresolution point of view, they provide powerful tools for speech and image coding applications. In this chapter we briefly discuss the basic wavelet theory and discrete wavelet transforms with orthogonal wavelets. We also explore the biorthogonal case in the last section.

B. WAVELET THEORY

Wavelet transforms can be considered an alternative to classical short-time Fourier transforms (STFT). The basic difference is that wavelet transforms use variable window sizes that change along the frequency range. The STFT analyzes signals by using a single window size [Ref. 1]; consequently, the time-frequency resolution is fixed over the entire time frequency plane (see Fig. 2.1(a)). Wavelet transforms provide an analysis method known as *constant-Q* or *constant-relative bandwidth analysis*. In this method we use a family of analysis filters where the time resolution increases with the center frequency of the filters. As shown in Fig. 2.1(b), wavelet analysis provides better frequency resolution at low frequencies and better time resolution at high frequencies. In other words, the time-frequency window is flexible such that it automatically narrows at a high

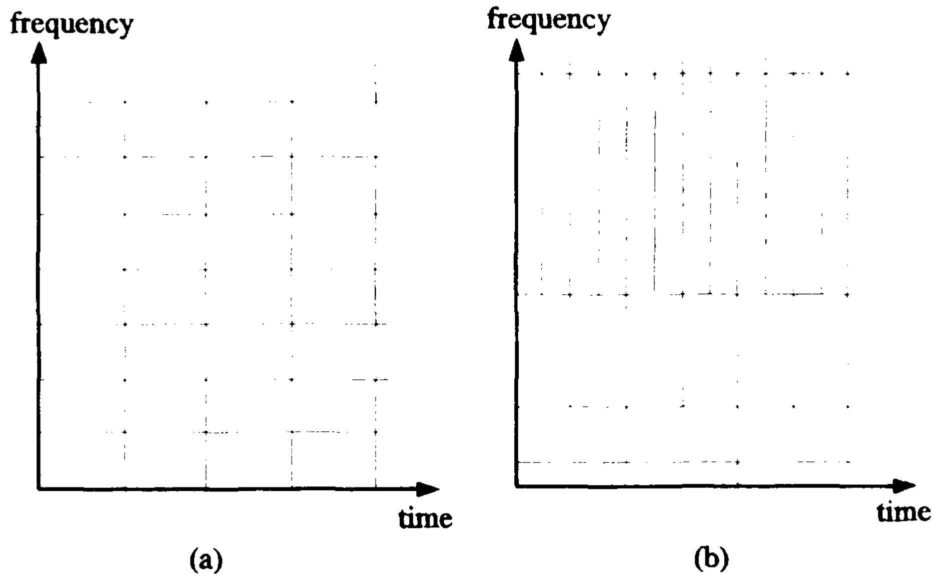

Figure 2.1 Coverage of Time-Frequency Plane for (a) STFT and (b) WT.

center frequency and widens at a low center frequency. However, the area under this window is independent of the center frequency.

In order to perform wavelet transforms, a set of basis functions called *wavelets* are required. These basis functions are obtained from a prototype function, $\psi(t)$, called the *mother wavelet*, by means of dilations and translations. The basis functions then take the form:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{|a|}}\psi\left(\frac{t-b}{a}\right) , \qquad (2.1)$$

where a and b are the dilation and translation parameters, respectively. A basis function becomes a dilated (low frequency) version of the prototype for a large value of a while it is a contracted (high frequency) version for a small value of a. The dilation and contraction of the mother wavelet in the time domain cause a corresponding contraction and dilation in the frequency domain.

The continuous wavelet transform (CWT) of x(t) is defined as the inner product of the form [Ref. 13]:

$$W_c(a,b) = \int x(t)\psi^*_{a,b}(t)dt \tag{2.2}$$

where the asterisk indicates complex conjugation. Since the $\psi_{a,b}$ are orthogonal, the original signal can be reconstructed by using the inverse relationship:

$$x(t) = c\int\int W_c(a,b)\psi_{a,b}(t)dadb, \tag{2.3}$$

where c is a constant of proportionality.

So far we have mentioned only the wavelet function, $\psi(t)$. For multiresolution analysis, we also need another basis function called the *scaling function*, $\phi(t)$. The scaling function is used to transform the signal from a fine to a coarse scale [Ref. 1]; moreover, its operation can be viewed as lowpass filtering while that of wavelet can be considered as highpass filtering. The scaling function has the form:

$$\phi_{a,b}(t) = \frac{1}{\sqrt{|a|}}\phi(\frac{t-b}{a}). \tag{2.4}$$

Equation 2.1 and 2.4 are similar in form.

1. Discretization of Dilation and Translation Parameters

The continuous wavelet transform has two drawbacks, namely, redundancy and impracticality. Both problems can be solved by sampling the dilation and translation parameters. This procedure leads to a family of wavelets and scaling functions with discrete parameters.

Let $a=a_0^m$ and $b=kb_0a_0^m$, then $\psi(t)$ and $\phi(t)$ defined in Eq. 2.1 and Eq. 2.4 become:

$$\psi_{mk}(t)=a_0^{-m/2}\psi(a_0^{-m}t-kb_0), \tag{2.5}$$

$$\phi_{mk}(t)=a_0^{-m/2}\phi(a_0^{-m}t-kb_0), \tag{2.6}$$

where m, k are integers and $a_0 > 1,\ b_0 \neq 0$. Inserting Eq. 2.5 into Eq. 2.2 yields wavelet coefficients in the discrete form [Ref. 5]:

$$W_d(m,k)=a_0^{-m/2}\int x(t)\psi(a_0^{-m}t-kb_0)dt. \tag{2.7}$$

The wavelet family is to be chosen so that the original signal can be uniquely reconstructed [Ref. 1]. In particular, they must form an orthonormal basis in $L^2(R)$, i. e., they must satisfy:

$$\int\psi_{mk}(t)\psi^*_{m'k'}(t)dt=\begin{cases}1, & m=m',k=k'\\ 0, & otherwise.\end{cases} \tag{2.8}$$

For the orthonormality condition, it can be shown that the parameter a_0 and b_0 must be chosen as $a_0 = 2$ and $b_0 = 1$ which imply the dyadic scale [Ref. 1].

2. Multiresolution Analysis

The idea of multiresolution analysis was introduced by Mallat [Ref. 3]. In his study, he used this approach to construct orthonormal wavelets. Assume that the original signal $x(t)$ is measurable and has finite energy. We successively decompose $x(t)$ into approximation and wavelet coefficients. This decomposition is done by convolving $x(t)$ with the scaling and wavelet filters.

Multiresolution analysis consists of a set of closed subspaces $\{V_m \mid 0 \geq m \geq M\} \in L^2(R)$, where we express the resolution level of the original signal at $m=0$ and the lowest resolution level at $m=M$ [Ref. 5]. The subspaces share certain properties:

- Causality : The vector space V_{m+1} of resolution level $m+1$ is a subset of the vector space V_m of the upper resolution level (m).
  $\leftarrow$ *coarser* $\ldots \subset V_{m+1} \subset V_m \subset V_{m-1} \subset \ldots$ *finer* $\rightarrow$

- Completeness: The vector spaces must satisfy the following operations:
  $\cup V_m = L^2(R)$ $\qquad\qquad$ $\cap V_m = \{0\}$.

- Scaling: The spaces of approximated signals can be computed by scaling each approximated signal by the ratio of their resolution values, i.e., if $x(t) \in V_m$, then $x(2t) \in V_{m-1}$.

- Orthogonality: $V_{m-1} = V_m \oplus W_m$, where W_m is the orthogonal complement of V_m in V_{m-1} , i.e., $V_m \perp W_m$. In other words, V_{m-1} is equivalent to V_m plus some added detail corresponding to W_m. Furthermore, this property and the causality property together imply that the direct sum of all W_m spans $L^2(R)$:

$$V_m = W_{m+1} \oplus W_{m+2} \oplus W_{m+3} \oplus \ldots \tag{2.9}$$

We associate the scaling function $\phi(t)$ with V_m, and the wavelet function $\psi(t)$ with W_m. We can write $\phi(t)$ as:

$$\phi(t) = 2\sum_n h(n)\phi(2t-n), \tag{2.10}$$

and similarly we can express $\psi(t)$ as:

$$\psi(t) = 2\sum_n g(n)\,\psi(2t-n), \tag{2.11}$$

where the coefficient set *h(n)* is the unit impulse response of a lowpass filter while *g(n)* is the unit impulse response of a highpass filter. The filter *g(n)* is derived from *h(n)* by:

$$g(n) = (-1)^n h(1-n). \tag{2.12}$$

The filters, *h(n)* and *g(n)*, must satisfy certain conditions for a perfect reconstruction, [Ref. 14]; namely:

$$\sum_n h(n) = 1 \tag{2.13}$$

$$\sum_n h^2(n) = \frac{1}{2} \tag{2.14}$$

$$2\sum_n h(n-2m)h(n-2k) = \delta(m-k). \tag{2.15}$$

The wavelet filter, *g(n)*, must also satisfy Eq. 2.14 and 2.15, but in place of Eq. 2.13, *g(n)* must satisfy:

$$\sum_n g(n) = 0. \tag{2.16}$$

From the orthogonality condition, *h(n)* and *g(n)* together satisfy the following equations:

$$2\sum_n [h(m-2n)h(k-2n) + g(m-2n)g(k-2n)] = \delta(m-k) \tag{2.17}$$

$$\sum_n h(n-2k)g(n-2m)=0. \tag{2.18}$$

Equation 2.10 through 2.18 describe the essential properties of these functions.

C. DISCRETE WAVELET TRANSFORMS

The discrete wavelet transform (DWT) was developed for processing discrete-time signals [Ref. 3, 4]. The DWT consists of a series of identical operation blocks with both time and dilation/translation parameters in discrete form. In this section, we start with the DWT for the 1-D case and then extend it to the 2-D case.

1. One-Dimensional DWT

The one-half resolution approximation of a discrete signal *x(n)* can be obtained by convolving it with a lowpass filter followed by subsampling by two. This forms the basic procedure for decomposing the original sequence into lower resolution levels. At the same time, convolving the sequence at each level with a highpass filter gives us the wavelet coefficients. Figure 2.2 depicts the pyramidal decomposition scheme of the DWT while Figure 2.3 shows the operations at a given resolution level. Each block in Fig. 2.2 performs the following operations:

- Convolution with the argument filter, *h(n)* or *g(n)*.
- Subsampling (decimation) by two.
- Normalization by $\sqrt{2}$.

The wavelet coefficients are the ones we need to save in order to reconstruct the original signal. We also need to save the approximation coefficients of the lowest level, *M*. In other words, the DWT representation of $x(n)$ consists of the coefficient set: $\{x_M, d_m, m=1,\dots,M\}$.

a. Decomposition

Given a discrete time sequence $x(n) \in L^2(R)$, the approximation coefficients are computed recursively from:

$$x_{m+1}(n)=\sqrt{2}\sum_k h(k-2n)x_m(k);\ m=0,1,\dots,M-1 \tag{2.19}$$

where $x_m(n)$ is the signal at resolution level m; $x_0(n)$ is the original signal. This operation is equivalent to lowpass filtering of $x_m(n)$ followed by subsampling by two. Consider an FIR structure for the filter, $h(n)$. The filtering operation in Eq. 2.19 can be represented as a matrix vector product:

$$\boldsymbol{x}_{m+1}=\sqrt{2}\boldsymbol{H}\boldsymbol{x}_m \tag{2.20}$$

where

$$\boldsymbol{H}=\begin{bmatrix} & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \ddots & h(L-1) & h(L-2) & \cdots & \cdots & h(0) & 0 & 0 & \ddots \\ & 0 & 0 & h(L-1) & \cdots & h(2) & h(1) & h(0) & \\ & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \end{bmatrix}. \tag{2.21}$$

and $\boldsymbol{x}_m = [\cdots x_m(-1), x_m(0), x_m(1), \cdots]^T$. The matrix, $\boldsymbol{H}$, must satisfy the condition [Ref. 4]:

$$2\boldsymbol{H}\boldsymbol{H}^T=\boldsymbol{I} \tag{2.22}$$

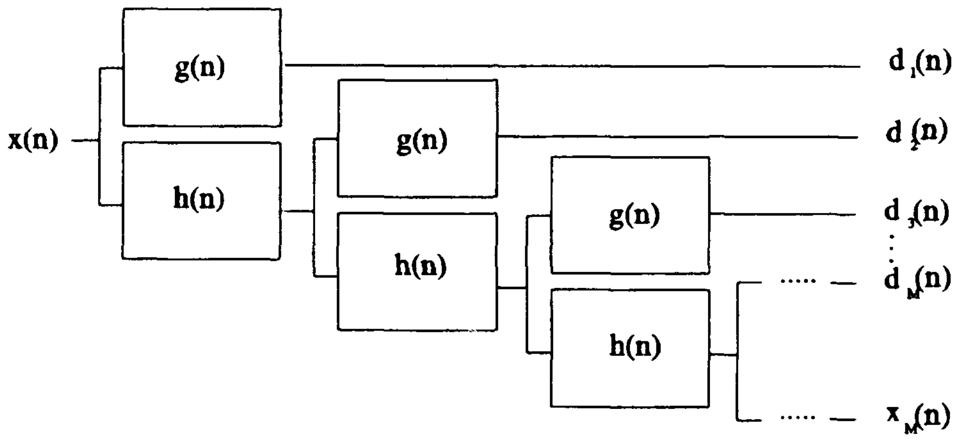

Figure 2.2 General Decomposition Algorithm of DWT

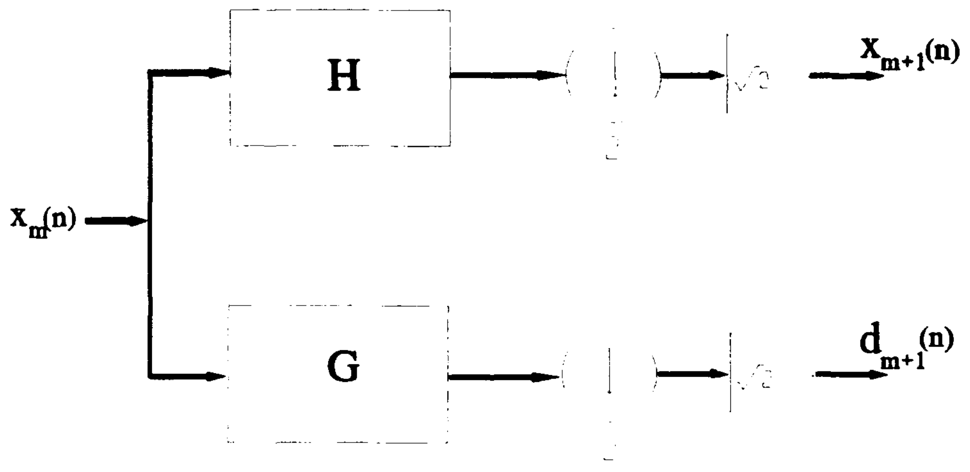

Figure 2.3 One-Step Decomposition Algorithm

where I is the identity matrix, and $(.)^T$ stands for the transposition operation. Notice the similarity between Eq. ? 22 and Eq. 2.14. Using the same algorithm with the wavelet filter gives us wavelet coefficients of lower resolution level:

$$d_{m+1}(n) = \sqrt{2}\sum_k g(k-2n)x_m(k), \tag{2.23}$$

where the coefficient set $g(n)$ is defined by Eq. 2.12. If we build a matrix, G, in a form analogous to Eq. 2.21, we have:

$$2GG^T = I, \tag{2.24}$$

and

$$HG^T = 0 \tag{2.25}$$

which follows from Eq. 2.18

b. Reconstruction

We know that the subspaces in which the signals are represented satisfy $V_{m-1} = V_m \oplus W_m$. This implies that we can reconstruct each higher level by adding the projections from the two orthogonal subspaces of the current level. The projections of a sequence onto V_m and W_m are given by:

$$\bar{x}_m = 2H^T H x_m \quad and \quad \bar{d}_m = 2G^T G x_m . \tag{2.26}$$

From Eq. 2.9 and 2.25 the projections are orthonormal and span complete subspaces. Thus, from Eq. 2.26 and the relation $\mathbf{x}_m = \bar{\mathbf{x}}_m + \bar{\mathbf{d}}_m$, we have:

$$2[H^T H + G^T G] = I \tag{2.27}$$

The reconstruction algorithm is depicted in Fig. 2.4 and has the form:

$$x_{m-1}(n) = \sqrt{2}\sum_k [h(n-2k)x_m(k) + g(n-2k)d_m(k)]. \tag{2.28}$$

This formula is derived as follows, substituting Eq. 2.19 and 2.23 into Eq. 2.28 and taking Eq.2.17 into consideration, we observe that the righthand side of Eq. 2.28 becomes:

$$\begin{aligned}\sqrt{2}\sum_k & [h(n-2k)x_m(k)+g(n-2k)d_m(k)] \\ &= \sqrt{2}\sum_k \{h(n-2k)[\sqrt{2}\sum_j h(j-2k)x_{m-1}(j)] + g(n-2k)[\sqrt{2}\sum_j g(j-2k)x_{m-1}(j)]\} \\ &= 2\sum_k \sum_j [h(n-2k)h(j-2k)+g(n-2k)g(j-2k)]x_{m-1}(j) \\ &= x_{m-1}(n).\end{aligned} \tag{2.29}$$

This completes the set of formulas needed for the one-dimensional DWT.

2. Two-Dimensional DWT

We now extend the one-dimensional DWT to the two-dimensional case. This can be done by using multidimensional extensions of wavelets. Assume that we have a set of subspaces V_m^1 satisfying the multiresolution conditions described in Section B.2. Then, we can define a vector space V_m as the tensor product of these subspaces in $L^2(R^2)$:

$$V_m = V_m^1 \otimes V_m^1. \tag{2.30}$$

If $\phi(t_l)$ is a one-dimensional scaling function of V_m^1, then we can derive the corresponding separable two-dimensional scaling function as:

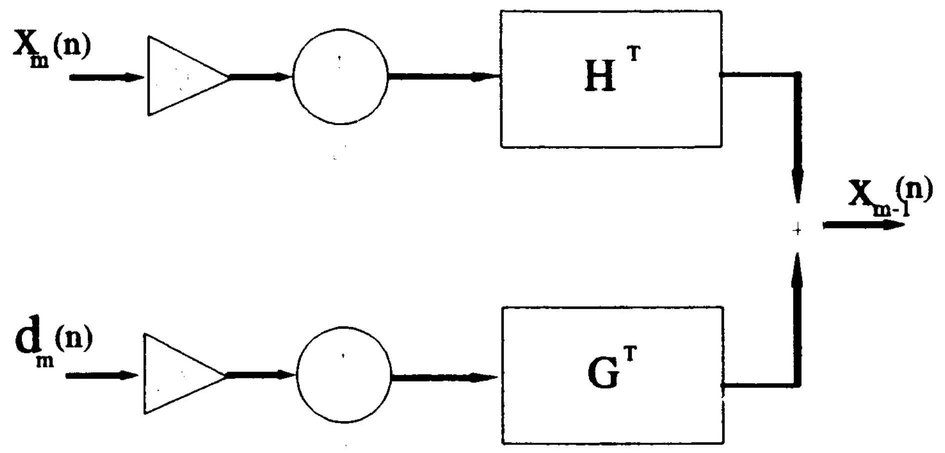

Figure 2.4 One-Step Reconstruction Algorithm.

$$\Phi(t_1,t_2)=\phi(t_1)\phi(t_2). \tag{2.31}$$

Similarly, two-dimensional wavelets can be obtained as:

$$\Psi^1(t_1,t_2)=\phi(t_1)\psi(t_2), \tag{2.32}$$

$$\Psi^2(t_1,t_2)=\psi(t_1)\phi(t_2), \tag{2.33}$$

$$\Psi^3(t_1,t_2)=\psi(t_1)\psi(t_2). \tag{2.34}$$

These functions are orthogonal to each other (follows from the orthogonality of one-dimensional components). Table 2.1 summarizes the two-dimensional basis functions and coefficients as well as their frequency characteristics. [Ref. 1, 3]

Table 2.1 SUMMARY OF BASIS FUNCTIONS.

Basis Functions	Coefficients	Frequency Characteristics
$\Phi(t_1,t_2)$	$x_m(n_1,n_2)$	Vertical Lowpass Horizontal Lowpass
$\Psi^1(t_1,t_2)$	$d^1_m(n_1,n_2)$	Vertical Lowpass Horizontal Highpass
$\Psi^2(t_1,t_2)$	$d^2_m(n_1,n_2)$	Vertical Highpass Horizontal Lowpass
$\Psi^3(t_1,t_2)$	$d^3_m(n_1,n_2)$	Vertical Highpass Horizontal Highpass

a. *Decomposition*

We can obtain the approximation, $x_m(n_1,n_2)$, and wavelet coefficients, $d^i_m(n_1,n_2)$, $i=1,2,3$, by convolving the approximation coefficients of higher resolution level by the horizontal and vertical scaling and wavelet filters:

$$x_{m+1}(n_1,n_2)=2\sum_k\sum_l x_m(k,l)h_h(l-2n_2)h_v(k-2n_1), \tag{2.35}$$

$$d^1_{m+1}(n_1,n_2)=2\sum_k\sum_l x_m(k,l)h_h(l-2n_2)g_v(k-2n_1), \tag{2.36}$$

$$d^2_{m+1}(n_1,n_2)=2\sum_k\sum_l x_m(k,l)g_h(l-2n_2)h_v(k-2n_1), \tag{2.37}$$

$$d^3_{m+1}(n_1,n_2)=2\sum_k\sum_l x_m(k,l)g_h(l-2n_2)g_v(k-2n_1). \tag{2.38}$$

The procedure for 2-D decomposition is depicted in Fig. 2.5. Here, we have two levels of 1-D decomposition: one is convolution with a horizontal basis filter in the horizontal direction (for example, rows of an image) and the second is convolution with a vertical basis filter in the vertical direction (for example, columns of an image).

b. Reconstruction

The reconstruction algorithm follows the same procedure of matrix multiplication in horizontal and vertical directions as in the decomposition scheme (see Fig. 2.6). In analogous to Eq. 2.25 and 2.27, we have:

$$2[H_v^T H_h^T H_h H_v + G_v^T H_h^T H_h G_v + H_v^T G_h^T G_h H_v + G_v^T G_h^T G_h G_v] = I. \tag{2.39}$$

The reconstruction algorithm in Fig. 2.6 can be expressed in matrix notation as:

$$x_m = 2H_v^T H_h^T x_{m+1} + 2G_v^T H_h^T d_{m+1}^1 + 2H_v^T G_h^T d_{m+1}^2 + 2G_v^T G_h^T d_{m+1}^3. \tag{2.40}$$

Substituting Eq. 2.35 through 2.38 into Eq. 2.40 and using the property of Eq. 2.39, we can show that:

$$\begin{aligned} &2H_v^T H_h^T x_{m+1} + 2G_v^T H_h^T d_{m+1}^1 + 2H_v^T G_h^T d_{m+1}^2 + 2G_v^T G_h^T d_{m+1}^3 \\ &\qquad = 2H_v^T H_h^T [H_h H_v x_m] + 2G_v^T H_h^T [H_h G_v x_m] \\ &\qquad\quad + 2H_v^T G_h^T [G_h H_v x_m] + 2G_v^T G_h^T [G_h G_v x_m] \\ &\qquad = x_m. \end{aligned} \tag{2.41}$$

The quality of reconstruction in both the 1-D and 2-D cases depends on the precision of the filter coefficients. If we use the Haar filter ($h=[0.5\ 0.5]$), we do not introduce any reconstruction errors. For other filters, such as Daubechies filters, it is possible to have some errors in the reconstructed signal because these filter coefficients require infinite precision [Ref. 2]. In practice, however, the resulting error is still insignificant (around 10^{-16} for our signals).

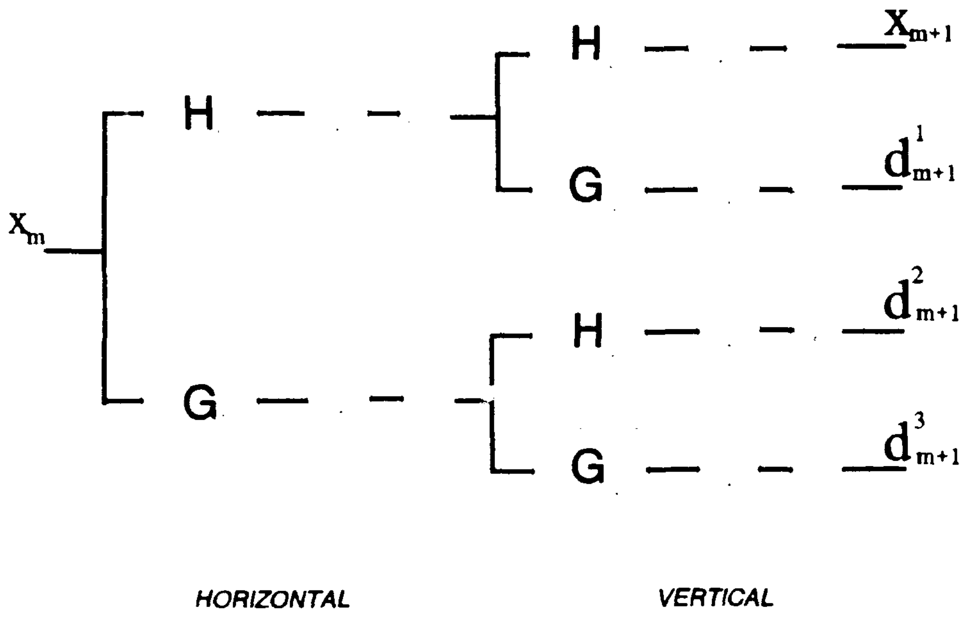

Figure 2.5 Decomposition Algorithm of 2-D DWT.

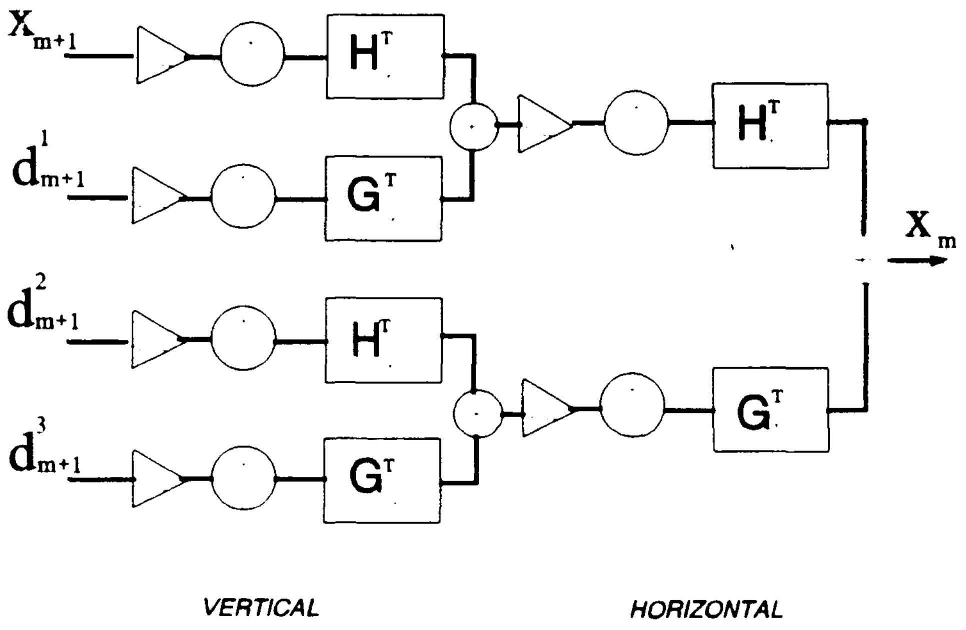

Figure 2.6 Reconstruction Algorithm of 2-D DWT.

D. BIORTHOGONAL WAVELETS

We prefer to use short filters for fast computation and because fewer redundant samples are caused by the convolution. Short filters do not provide adequate smoothness however. For image analysis, linear phase filters are preferred [Ref. 4]. In order to have filters of arbitrary length with linear phase, we have to sacrifice orthogonality [Ref. 4]. The solution to this problem is to use filters based on a special class of functions called *biorthogonal* bases.

Biorthogonality requires use of different filters for analysis and synthesis operations with the decomposition and reconstruction algorithms described in Section B of this chapter. Figure 2.7 shows a one step decomposition and reconstruction scheme using biorthogonal filters. The conditions for perfect reconstruction are satisfied by imposing orthogonality separately over the decomposition and reconstruction stages [Ref. 4]. Using the matrix notation we have used in Eq.2.21, we have:

$$\boldsymbol{H}_0\boldsymbol{G}_1=\boldsymbol{G}_0\boldsymbol{H}_1=\boldsymbol{0}, \tag{2.42}$$

$$2\boldsymbol{H}_0\boldsymbol{H}_1=2\boldsymbol{G}_0\boldsymbol{G}_1=\boldsymbol{I}. \tag{2.43}$$

Given the analysis and synthesis scaling filters, $h_0(n)$ and $h_1(n)$, the corresponding wavelet filters are expressed as:

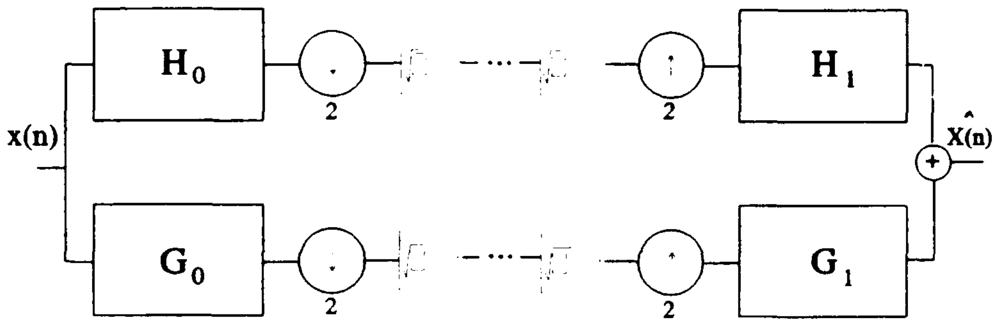

Figure 2.7 DWT with Biorthogonal Wavelets.

$$g_0(n) = (-1)^n h_1(1-n), \tag{2.44}$$

$$g_1(n) = (-1)^n h_0(1-n). \tag{2.45}$$

Notice the similarity between Eq. 2.25 and Eq. 2.42, and Eq. 2.22 and Eq. 2.43 (in the orthogonal case $H_1 = H_0^T$ and $G_1 = G_0^T$). Finally, perfect reconstruction in biorthogonal bases requires:

$$2[\boldsymbol{H}_1 \boldsymbol{H}_0 + \boldsymbol{G}_1 \boldsymbol{G}_0] = \boldsymbol{I}. \tag{2.46}$$

The biorthogonal DWT algorithm for one-dimensional signals can be easily extended to the two-dimensional case by using biorthogonal filters in the vertical and horizontal directions.

III. VECTOR QUANTIZATION

A. INTRODUCTION

Vector quantization (VQ) is generalization of scalar quantization where a scalar value is represented by one of several discrete levels. In vector quantization a vector quantity is represented by one of several other fixed vectors which are close to the original vector in some sense. According to Shannon's rate distortion theory, better results are always obtained when vectors instead of scalars are encoded [Ref. 15]. As the vector dimension becomes large, VQ can approach the rate distortion limit for the problem. In typical applications, bit rates of lower than one bit per sample are achievable with vector quantization; these rates are unlikely for scalar quantization.

B. QUANTIZER DESIGN

Vector quantizers map vectors in a multidimensional space into a finite set of reproduction vectors called the *codebook*. The codebook is then used to encode and decode the data set. Figure 3.1 shows a schematic of a basic vector quantizer. The encoder quantizes blocks of the discrete time signal *x(n)* by using the codebook constructed previously and encodes it into a sequence of bits $\{b_i\}$, which is then transmitted through the transmission channel or stored in some storage medium. If there are no channel errors, then $\{b_i'\}=\{b_i\}$. At the receiver end, a decoder converts the sequence of bits $\{b_i'\}$ into a vector representing blocks of data and thus into the signal,

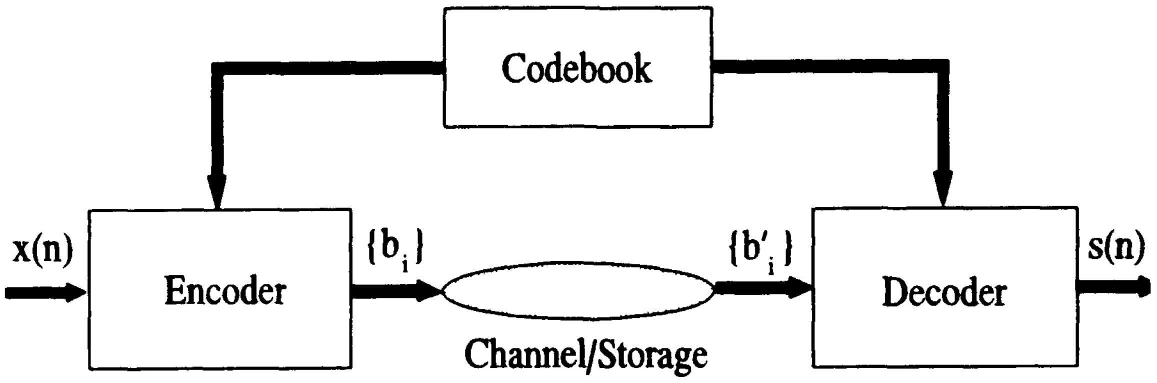

Figure 3.1 Basic Structure of Vector Quantization.

s(n). The output *s(n)* is the reconstructed signal, which will be an approximation of the input signal *x(n)*.

Given the basic structure of VQ, we now proceed to discuss the details of VQ. Assume that $\boldsymbol{x}=[x_1, x_2, \ldots, x_N]^T$ is an N-dimensional vector whose components $\{x_k, 1\leq k\leq N\}$ are real and continuous (amplitude) random variables. Specifically, the x_k represent data samples in a block of data from the signal *x(n)*. A vector quantizer maps $\boldsymbol{x}$ into another real-valued, discrete-amplitude, N-dimensional vector $\boldsymbol{y}$. This operation can be described as:

$$y=q(x), \tag{3.1}$$

where *q(.)* is the quantization operatoı. The reproduction vector y is chosen from a set of vectors $M=\{y_i, 1\leq i\leq L\}$, where $y_i=[y_{i1}, y_{i2}, \ldots, y_{iN}]^T$. The set M is called the *codebook*, where L is the size of the codebook. The quantity

$$R = \log_2 L \tag{3.2}$$

is the *rate* of the quantizer in bits per vector while $r=R/N$ is the rate in bits per sample (or bits per pixel if the input is image data). To design a codebook, we partition the N-dimensional space of input vectors $\boldsymbol{x}$ into L cells $\{C_i,\ 1 \le i \le L\}$ and associate a vector $\boldsymbol{y}_i$ with each cell C_i. Then we use the code vector $\boldsymbol{y}_i$ to represent vectors that fall within cell C_i, that is:

$$q(x) = y_i, \qquad if\ x \in C_i. \tag{3.3}$$

The vector quantizer is said to be a minimum-distortion quantizer if some measure of distortion is minimized over all code vectors. The measure of distortion is defined in terms of a distortion function, $d(x,y)$, which represents the cost of representing any input vector $\boldsymbol{x}$ by a code vector $\boldsymbol{y}$. Although there are many possible distortion measures [Refs. 6, 8], we choose the *mean-square error* (MSE):

$$d(x,y) = \frac{1}{N}\sum_{k=1}^{N} (x_k - y_k)^2 \tag{3.4}$$

due to its simplicity and mathematical tractability. This distortion measure is simply the squared Euclidean distance between **x** and **y**.

To design an optimal (minimum-distortion) quantizer using this distortion function, there are two necessary conditions [Ref. 7]. First, the quantizer must satisfy the nearest neighbor condition:

$$q(x) = y_i, \qquad \text{iff } d(x,y_i) \leq d(x,y_j),\ j \neq i,\ 1 \leq j \leq L. \tag{3.5}$$

That is, for each i, all input values that are closer to code vector y_i than to any other code vector should be assigned to cell C_i. Therefore:

$$d(x,q(x)) = \min_{y_i \in M} \{d(x,y_i)\}. \tag{3.6}$$

Secondly, the code vectors y_i must satisfy the centroid condition; each code vector y_i is chosen such that y_i minimizes the MSE in cell C_i. In this case, it is easily shown that:

$$y_i = cent(C_i). \tag{3.7}$$

where:

$$cent(C_i) = \frac{1}{K}\sum_{k=1}^{K} x_k \tag{3.8}$$

for all of the vectors $\{x_k,\ k=1,\ 2,\ .\ .\ .\ .\ ,\ K\}$ contained in cell C_i.

Based on the conditions defined above, an iterative procedure can be defined to optimize the quantizer starting from an arbitrary set of code vectors. The iteration takes place until convergence is obtained. The algorithm used is the *generalized Lloyd (GL) algorithm* [Ref. 8] also known as the LBG (Linde,Buzo,Gray) algorithm [Ref. 9]. The algorithm is detailed as follows :

- **Step 1** (Initialization): Set $m=1$. Choose an initial codebook M_1.
- **Step 2** (Classification): Given the codebook M_1, classify the set of training vectors $\{x_k,\ 1 \leq k \leq L\}$ into cells C_i according to the nearest neighbor condition.
- **Step 3** (Updating): $m \leftarrow m+1$. Update the code vector y_i of each cell C_i by the centroid condition (see Eq. 3.7).

- **Step 4** (Termination Test): Compute MSE for M_{m+1}. If it has changed by an amount less than a prespecified value, stop; otherwise, go to Step 2.

A flow chart for this algorithm is shown in Fig. 3.2.

The algorithm above requires an initial codebook for the first step. There is a variety of techniques available for generating the initial codebook (see Refs. 6 and 8 for different methods). In this work, we choose the *random coding* technique. Given a set of training vectors, this method randomly selects L code vectors. Once the initial code vectors are selected, the GL algorithm is performed to improve the codebook. If an initial code vector y corresponds to a cell where there are no training vectors, the empty cell problem arises. In this thesis we assume that a cell is empty if it has three or fewer training vectors. If an empty cell arises, we split the code vector with the highest number of training vectors into two code vectors for that particular cell and repartition the cell into two cells. The empty cell is then discarded. This algorithm forms Step 2 (classification) of the GL iteration (see Fig. 3.3). The criterion to terminate the iteration in Step 4 (Figure 3.2) is based on the change in MSE, which is computed as:

$$\Delta(MSE) = \frac{D_m - D_{m+1}}{D_m}. \qquad (3.9)$$

where $D_m = d(x,y)$ for iteration m.

The quantization procedure described here is known as *full search VQ* (FSVQ) since all code vectors are tested for quantizing each input vector. Other methods of VQ exist (see Refs. 6 and 8), but were not explored in this thesis.

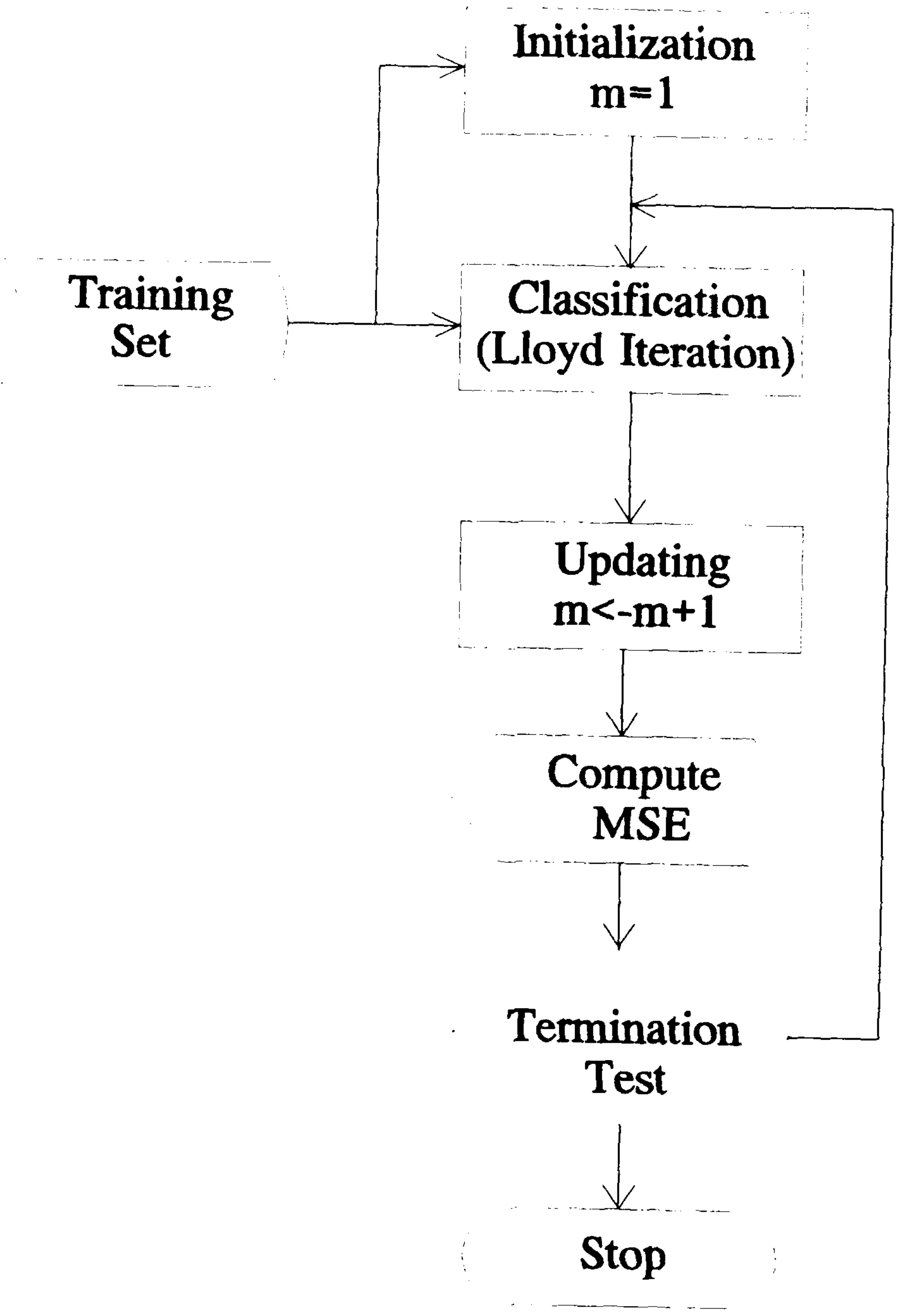

Figure 3.2 Flow Chart for Generalized Lloyd Algorithm.

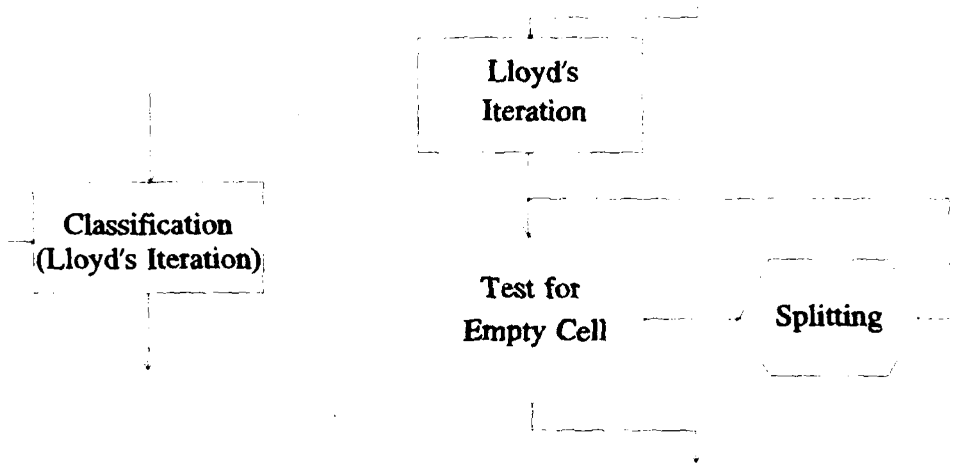

Figure 3.3 Modification of GL Algorithm to Avoid the Empty Cell Problem.

IV. ALGORITHM DEVELOPMENT FOR SPEECH DATA AND ANALYSIS

A. INTRODUCTION

The previous two chapters provide an overview of discrete wavelet transform and vector quantization. In this chapter we combine these concepts to develop algorithms for speech waveform compression. We also include a test case for transient signal compression in the last section.

The method used here is known as *analysis-synthesis coding*. It has been commonly used in subband coding and other transform coding techniques [Ref. 8, 16]. This method analyzes the signal into components that in some sense provide a more primitive representation of the signal, and quantizes these components. The quantized components are then used to synthesize a reproduction of the original signal. Figure 4.1 illustrates the basic steps of this algorithm. Notice that Figure 4.1 is a modified version of Fig. 3.1, where we directly quantize the original signal without decomposing it into components.

In this study, the DWT is used for analysis-synthesis operations while vector quantization is the method chosen for encoding the wavelet coefficients. The DWT provides a set of wavelet coefficients for each resolution level. Each set of coefficients has different statistical characteristics, and by designing a quantizer for each coefficient set optimally, the quantization error can be reduced. This error reduction at each level results in an improvement in the quality of reproduction of the original data.

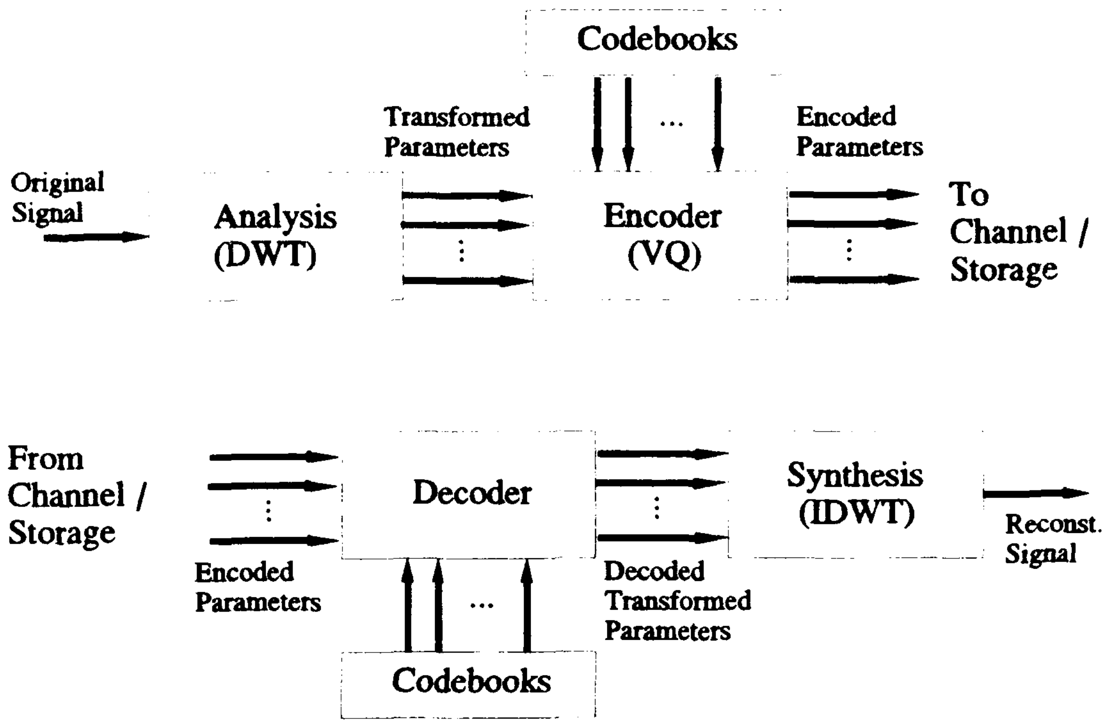

Figure 4.1 Analysis-Synthesis Coding Scheme.

In the following sections we develop algorithms for speech data based on the analysis-synthesis coding scheme. We also present the experimental results of the algorithm and comparisons of this method with direct vector quantization (DVQ).

B. ALGORITHM DEVELOPMENT

The analysis stage for speech data consists of a four-level DWT decomposition. Here, DWT decomposes the speech data into four sets of wavelet coefficients, $\{d_m, m=1,\dots, 4\}$, and one set of approximation coefficients, x_4 defined with Eq. 2.23 and Eq. 2.19, respectively. From this point on we call the combined set of coefficients $\{x_4,$ and $d_m, m=1,\dots, 4\}$ the *wavelet coefficients* for convenience. Figure 4.2 shows the

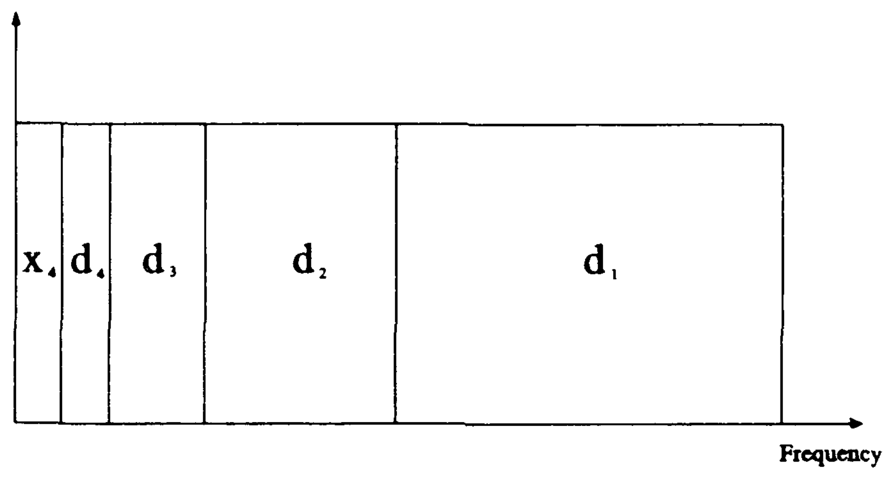

Figure 4.2 Division of Frequency Domain for DWT.

division of the frequency spectrum for each resolution level. At each resolution level, the lower band is divided into low and high frequency bands and subsampled in the time domain by two as explained in Chapter II, Section C. The scaling filter, *h(n)*, that we have chosen for the DWT is Daubechies 10-tap filter listed in Ref. 2. The wavelet filter, *g(n)*, is derived from the scaling filter by Eq. 2.12.

The next step after obtaining the wavelet coefficients is to construct the codebooks for each set. The sizes of the codebooks are determined by the number of bits allocated for each codebook. The bit allocation is a major concern in the design of a coding system. It consists of distributing a given quota of bits to various quantizers in order to optimize the overall coder performance [Ref. 8].

1. Optimum Bit Allocation

Fewer bits can be allocated to the high frequency components than to the low frequency components since the corresponding coefficients usually have a smaller variance. It is also possible to discard all coefficients at a given resolution level, *m*, if the normalized energy:

$$E(d_m) = \frac{\sum_n d_m^2(n)}{\sum_n x^2(n)} \tag{4.1}$$

in that level is less than a predetermined threshold value. The threshold value is determined such that a coefficient set with low energy level does not have a significant contribution to the reproduction of the original data. Discarding some coefficient sets by this approach allows us to allocate more bits to encode the coefficients at other levels with significant energy values.

While the energy $E(d_m)$ is used as a threshold to select the desired wavelet coefficient sets, the variance of the corresponding wavelet coefficients is used in bit allocation calculations. Let B represent the desired average number of bits per vector, and σ_m^2 be the variance of the wavelet coefficients at resolution level m. Then the optimal bit assignment is given by [Ref. 8]:

$$b_m = \frac{B + \frac{1}{2}\log_2 \frac{\sigma_m^2}{\rho^2}}{W_m}, \tag{4.2}$$

where

$$\rho^2 = (\prod_{m=1}^{K} \sigma_m^2)^{1/k} \tag{4.3}$$

is the geometric mean of the variances of the wavelet coefficients, and the parameter W_m is defined as the ratio of the number of coefficients at resolution level m, L_m, to the number of samples in the original signal, L_0:

$$W_m = \frac{L_m}{L_0}. \tag{4.4}$$

The number of coefficients, L_m, at resolution level m is given as:

$$L_m = \left\lfloor \frac{L_{m+1} + F - 1}{2} \right\rfloor, \tag{4.5}$$

where F is the filter length, and $\lfloor . \rfloor$ represents the floor operation.

In practice, this bit allocation method does not guarantee integer bit assignments; moreover, it does not exclude the possibility of negative bit allocation. We eliminate the negative bit allocation problem by replacing negative values by zero. Noninteger bit assignments are adjusted to the nearest integer.

2. Vector Quantization of Wavelet Coefficients

The energy distribution of speech data over the wavelet coefficients has certain similarities. For a four-level DWT decomposition, the energy is typically concentrated in the wavelet coefficients at resolution levels 3 and 4, d_3 and d_4, while the wavelet coefficients d_1 and x_4 have relatively less energy. Figure 4.3 shows the energy

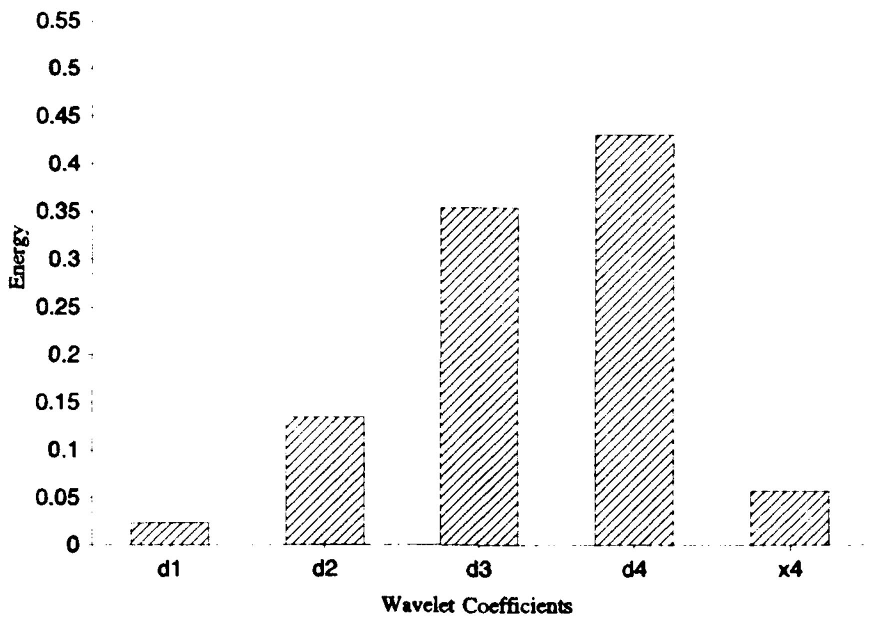

Figure 4.3 Energies of Wavelet Coefficients.

distribution of wavelet coefficients of a sample speech data. As the energy levels of d_1 and x_4 are low compared to those of d_3 and d_4, we discard the wavelet coefficients d_1 and x_4 by replacing them with zeros. (The threshold was set at $E(d_m)=0.06$.) The quality of the reproduced speech indicates that these coefficients (d_1 and x_4) do not have a significant contribution in the reproduction.

Having determined the coefficients to encode, there is another problem to be resolved: the small size of training sets at lower resolution levels. Equation 4.5 indicates that the total number of samples decreases by approximately a factor of two with each resolution level; furthermore, this implies that the size of the training set for that particular resolution level shrinks accordingly. One way to compensate for this effect is to use a large number of samples in the original data. However, in some cases, such large data sets may not be available. Instead, we use the following technique to extend the data sets at low resolution levels. Consider the resolution level *m*, where we have roughly half the number of samples as at resolution level *m-1* after carrying out the DWT. We now shift the data at level *m-1* by one sample and repeat the DWT decomposition for the shifted data. This gives us a second set of coefficients for resolution level *m*. By combining the two sets of coefficients, we double the number of coefficients at resolution level *m*. The resulting increase in the training set size offsets the decrease caused by subsampling.

The next step is to construct the codebooks by using the training data sets obtained by the procedure outlined above. The sizes of codebooks are determined by the bit allocation algorithm described in Eq. 4.2. The wavelet coefficients at selected resolution levels are then encoded by using the corresponding codebooks. In order to reconstruct the original data, we decode the wavelet coefficients by using the same codebooks and take the inverse DWT. Here, we substitute zeros for the discarded wavelet coefficients.

C. SIMULATION RESULTS

In this section we study the performance of the wavelet transform coding (WTC) using speech data. Simulations were carried out by using two data sets:

- A training sequence of 104100 samples of ordinary speech from three different speakers sampled at 8 kHz.
- A test sequence of 20330 samples from a speaker not in the training sequence.

The test sequence was also encoded by DVQ for comparison. We kept the average bit allocation at $B=2$ bits per sample (bps) (see Eq. 4.2) and changed the vector size to obtain different bit rates. The codebook sizes for wavelet coefficients are listed in Table 4.1. The distortion criterion defined in Eq. 3.9 for the GL algorithm was set at 0.01. The performance of the quantization was measured by using a signal-to-noise ratio (SNR) defined as:

$$SNR = 10\log_{10}\frac{E(x^2)}{D} \quad dB \tag{4.6}$$

where $E(x^2)$ is the signal power and D is the mean-square error between the original and reconstructed signal. The signal power is the estimated variance, σ_x^2, of signal samples assuming that the input has zero mean.

Table 4.1 CODEBOOK SIZES FOR WAVELET COEFFICIENTS

	Wavelet Coefficients				
	d_1	d_2	d_3	d_4	x_4
Codebook Size	0	8	64	512	0

The performance of WTC in terms of SNR vs. bit-rate (bits per sample) is plotted in Fig. 4.4 as the solid line. The dotted line represents the performance of DVQ. Clearly, WTC is superior to DVQ with a SNR improvement of between 19% and 55%, from 0.18 bps to 1.0 bps. Figure 4.5 shows a segment of original test sequence. Figure 4.6 shows an encoded example with a bit rate of 0.18 bps and an SNR of 3dB. The overall quality is fairly good and the speech is understandable. The same speech

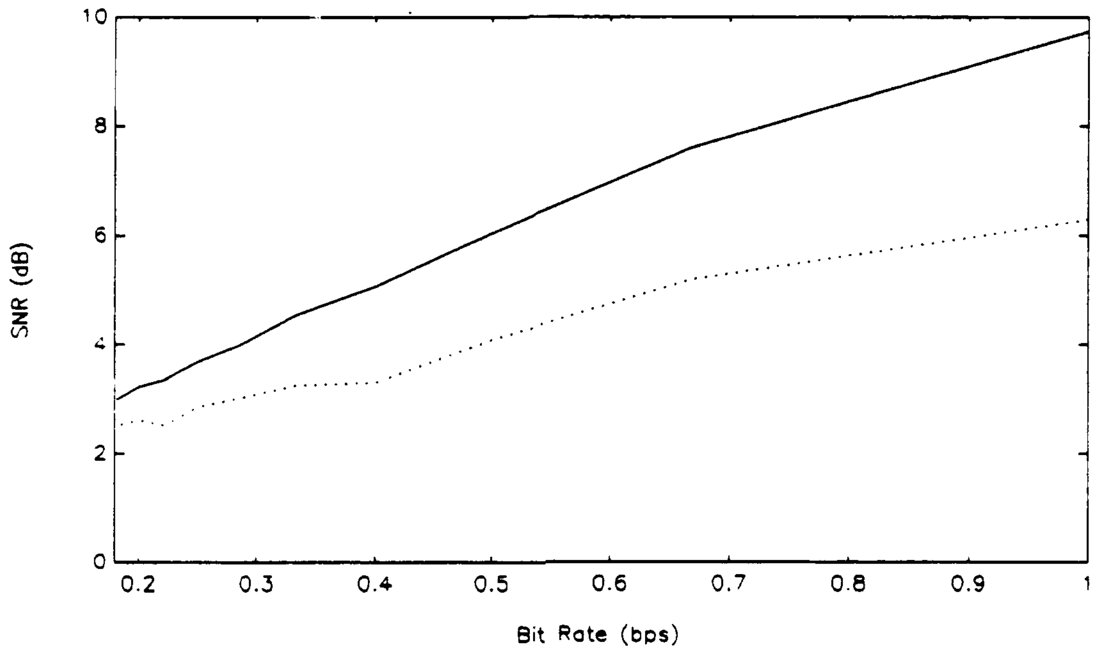

Figure 4.4 SNR vs. Bit Rate: Speech Data. The solid line represents WTC while the dotted line represents DVQ.

segment encoded by DVQ is shown in Fig. 4.7. The bit rate was 0.18 bps and SNR was 2.5dB. Although the SNR values of WTC and DVQ are fairly close, the smoothing capability of WTC renders the speech more understandable. Figures 4.8-4.9 show the reproduced speech segments encoded at bit rates of 1.03 bps for WTC and 1 bps for DVQ. Table 4.2 summarizes the bit rates and the contribution of each wavelet coefficient set to the overall distortion.

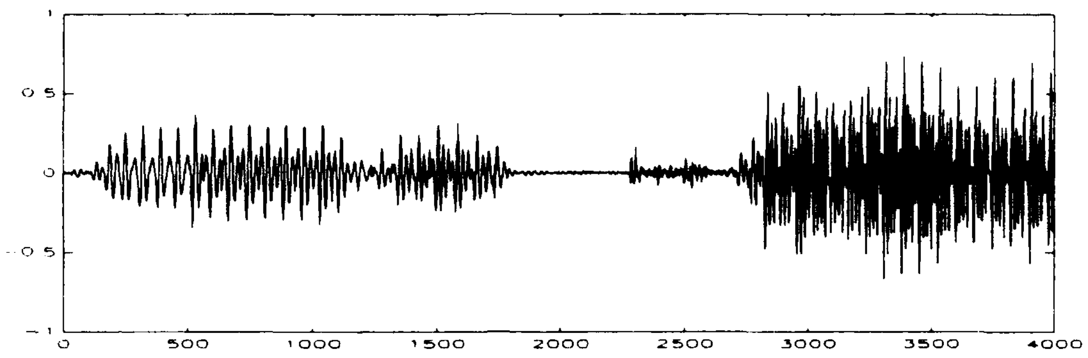

Figure 4.5 Original Test Speech.

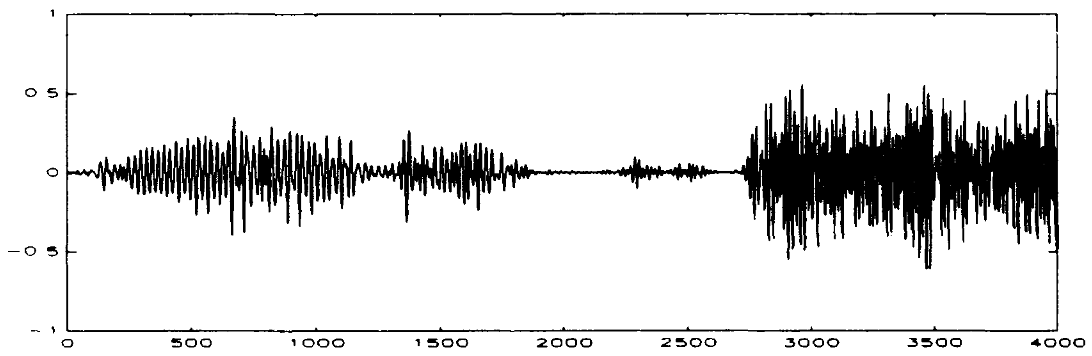

Figure 4.6 Test Speech Coded by WTC at 0.18 bps, SNR = 3.0 dB.

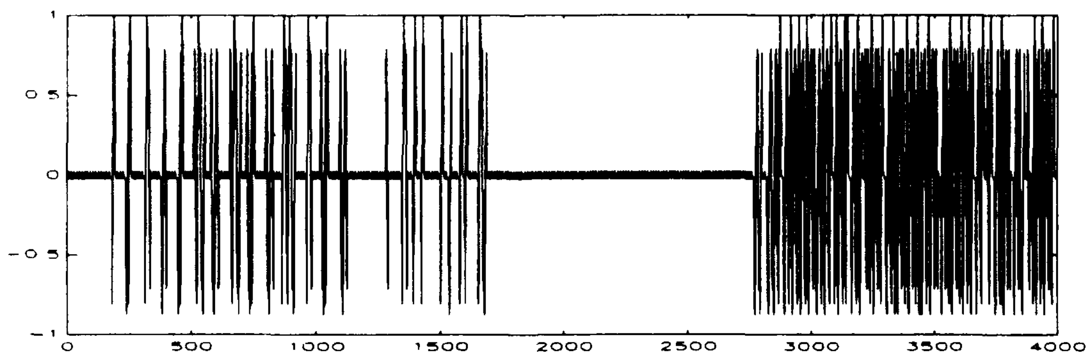

Figure 4.7 Test Speech Coded by DVQ at 0.18 bps, SNR = 2.5 dB.

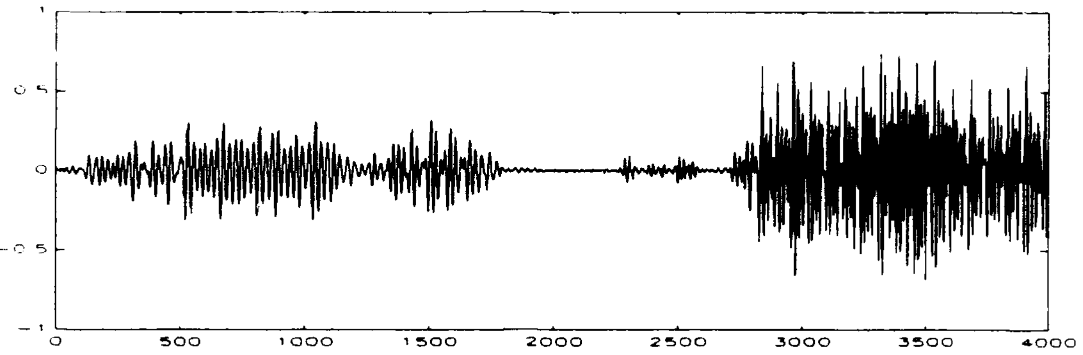

Figure 4.8 Test Speech Coded by WTC at 1.03 bps, SNR = 9.56 dB.

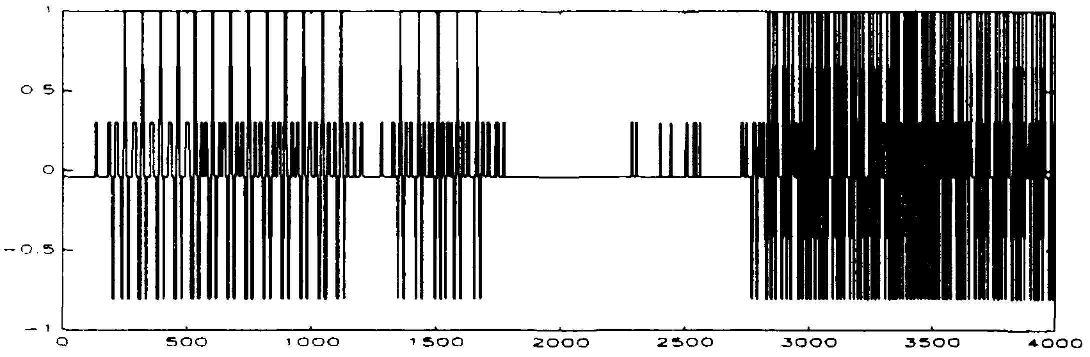

Figure 4.9 Test Speech Coded by DVQ at 1.0 bps, SNR = 6.2 dB.

Table 4.2 BIT ALLOCATIONS AND MSE CONTRIBUTIONS OF WAVELET COEFFICIENTS FOR AN AVERAGE BIT RATE OF 1.03 BPS.

	Wavelet Coefficients				
	d_1	d_2	d_3	d_4	x_4
Bit Rate (bps)	0	0.3760	0.3774	0.2853	0
MSE	0.0006	0.0021	0.0067	0.0020	0.0154

Interestingly, given an average number of bits, B, the performance of WTC increases rapidly over the performance of DVQ when we use smaller vector dimensions. (Recall that we use larger vectors to obtain small bit rates in Fig. 4.4.) This is caused by the effect of correlation between samples. At lower levels, the wavelet coefficients become less correlated and result in less degradation of quality for small vector dimensions.

We have, additionally, tested the WTC algorithm for coding transient signals. For the simulation study, a transient signal of 2400 samples was utilized. Since the energy levels of the wavelet coefficients, x_4 and d_4, were low compared to those of d_1, d_2 and d_3, a three-level DWT decomposition was chosen to analyze the original signal. Among the four sets of wavelet coefficients (x_3 and d_i, $i=1, 2, 3$), x_3 was discarded due to its low energy level (see Table 4.3), and the bit allocation algorithm in Eq. 4.2 was applied to the coefficients, d_1, d_2 and d_3, keeping the average number of bits at $B = 2$. Table 4.3 shows the codebook sizes for the different resolution levels as well as the energy levels of the wavelet coefficients at those resolution levels.

Table 4.3 ENERGY LEVELS AND CODEBOOK SIZES OF WAVELET COEFFICIENTS OF TRANSIENT SIGNAL

	Wavelet Coefficients			
	d_1	d_2	d_3	x_3
Energy	0.0690	0.8152	0.1030	0.0140
Codebook Size	2	16	8	0

As in the case of speech waveform coding, the test signal was coded by both WTC and DVQ to allow good comparisons. The vector dimensions were changed to achieve varying bit-rates in terms of bits per sample. The original signal is shown in Fig. 4.10. Figure 4.11 shows the signal coded by WTC at 1.0 bps. The same test signal coded by DVQ at 1.0 bps is displayed in Fig. 4.12. The SNR of the WTC-coded signal was 7.1 dB while the SNR in case of DVQ was only 3.9 dB. The WTC reproduces the transient signal better because it decomposes the original signal into different resolution levels corresponding to different frequency bands. Also, the DVQ does not contain a sufficient number of code vectors to represent the changing structure of the transient signal well. The improvement realized by WTC is confirmed by the SNR performance displayed in Fig. 4.13.

For the two test cases (speech waveform and transient signal coding), the improvement in the performance of WTC over DVQ in terms of SNR varied from 19 to 84 percent. As stated before, this advantage is a result of utilizing different codebooks to quantize the wavelet coefficients at different resolution levels and also of discarding components that do not contribute to the reproduction of the original signal.

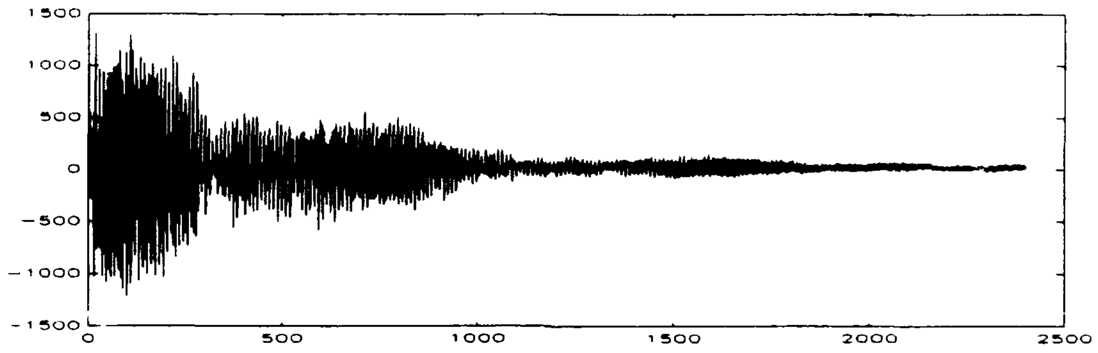
Figure 4.10 The Original Transient Signal.

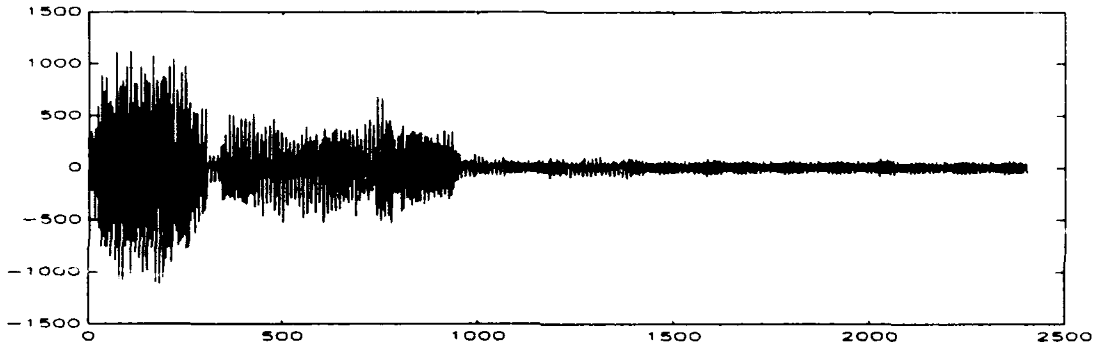
Figure 4.11 Transient Signal Coded by WTC at 1.0 bps, SNR = 7.1 dB.

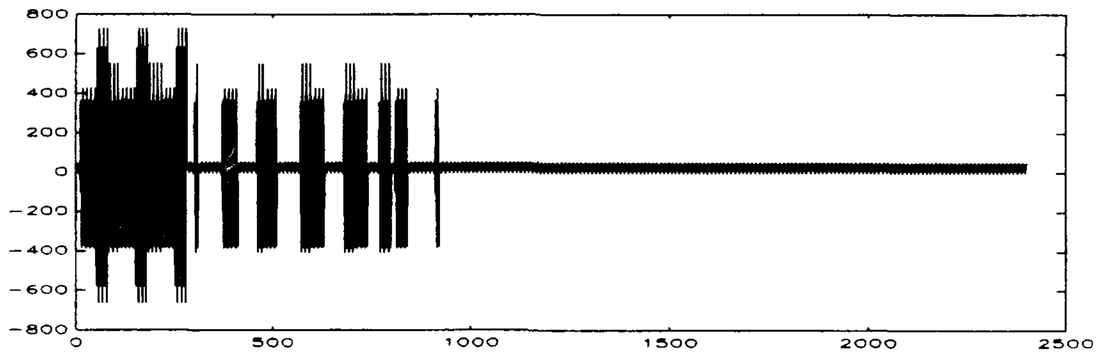
Figure 4.12 Transient Signal Coded by DVQ at 1.0 bps, SNR = 3.9 dB.

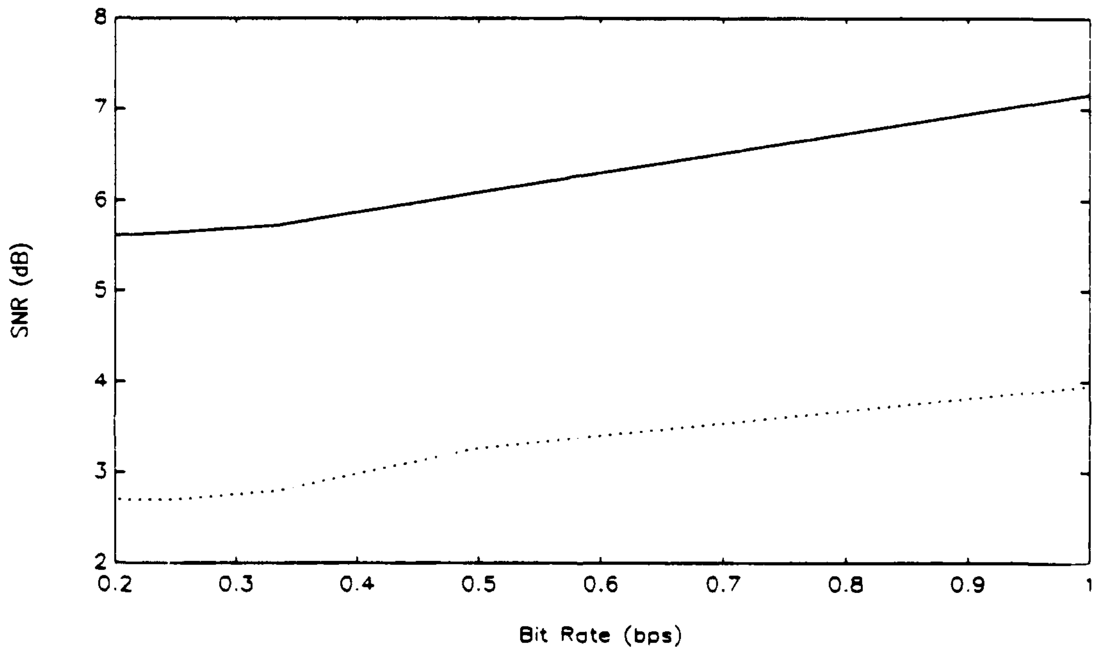

Figure 4.13 SNR vs. Bit Rate: Transient Signal. The Solid Line Represents the WTC, and the Dotted Line Represents DVQ.

V. ALGORITHM DEVELOPMENT FOR IMAGE DATA AND ANALYSIS

A. INTRODUCTION

In this chapter we extend the algorithm developed in the previous chapter to the two-dimensional case. The algorithm consists of the same operations illustrated in Fig. 4.1. The following sections provide a detailed overview of the algorithm and the simulation results.

B. ALGORITHM DEVELOPMENT

The analysis step of the proposed coding method includes a three-level DWT decomposition in two-dimension as illustrated in Fig. 5.1. A three-level instead of a four-level decomposition as used in the speech waveform coding was chosen because the correlation between coefficients of an image at lower resolution levels decreases significantly, and this makes encoding the wavelet coefficients at those levels extremely difficult. Table 5.1 shows the variances of the wavelet and approximation coefficients at different resolution levels. Notice the increase in variance of x_4 at the fourth resolution level. One way to compensate the effect of increase in variance would be to use more bits in that particular resolution level. But, using more bits increases the computational cost involved in generating the codebook and finding the closest codeword for each vector; thus, we stop the decomposition at the third resolution level.

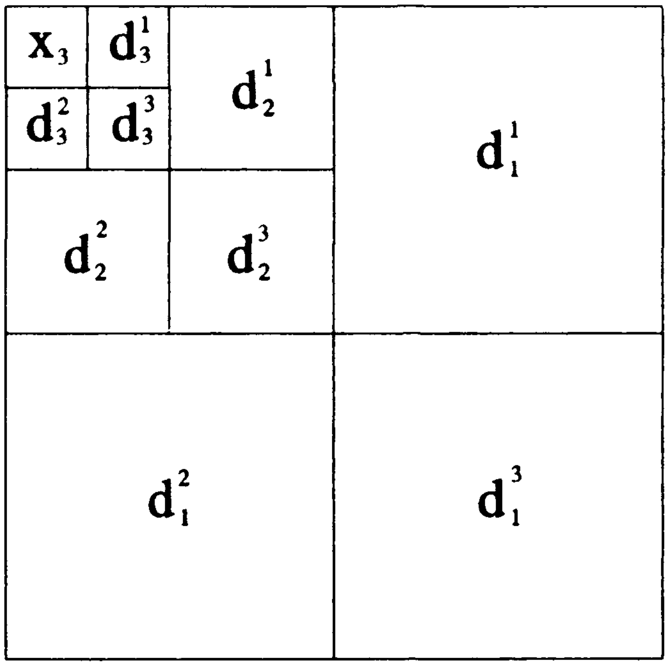

Figure 5.1 Three-level DWT Decomposition.

Table 5.1 VARIANCES OF WAVELET COEFFICIENTS (10^4)

	Resolution Levels			
	1	**2**	**3**	**4**
x	0.8504	4.4047	23.5356	116.6700
d_1	0.0126	0.0495	0.4595	4.2170
d_2	0.0050	0.0439	0.3668	2.5067
d_3	0.0008	0.0073	0.0674	0.4239

As we mentioned in Section D of Chapter II, we would like to have short wavelet filters, and we also prefer to use filters with linear phase characteristics. By relaxing the orthogonality and choosing biorthogonal basis functions, it is possible to use linear phase filters. Throughout the following experiments, we used the 7-tap biorthogonal filters listed in Table 5.2 which are short and have linear phase.

Table 5.2 7-TAP BIORTHOGONAL FILTER COEFFICIENTS

n	Analysis Filter (h_0)	Synthesis Filter (h_1)
0	0	0.607142857143
± 1	-0.05	0.260714285714
± 2	0.25	-0.053571428571
± 3	0.60	-0.010714285714

1. Optimum Bit Allocation

The three-level DWT provides 10 sets of wavelet coefficients $\{x_3, d^i_m, m, i = 1, 2, 3\}$ for quantization. Figure 5.2 shows the average energy distribution of wavelet coefficients of sample images; the energy of x_3 is not shown to scale in order to make other energy bars visible. (The actual value for x_3 is shown on top of the corresponding bar.) As can be seen, most of the energy is concentrated in x_3; the energies contained in the other coefficients are much smaller. Using the bit allocation scheme defined in the previous chapter leads to a problem because of this uneven energy distribution and the high variance of x_3 (See Table 5.1); all bits are assigned to wavelet coefficients x_3 leaving no bits to represent the other wavelet coefficients. If we discard the other coefficients and encode only x_3, this causes another problem. The edge information of the image is mostly contained in these coefficients, and the edges in the image are lost if we discard all of these coefficients. In order to allocate some bits to quantize d^i_m, we first reserve a large number of bits for x_3 taking the computational cost into account and share the remaining

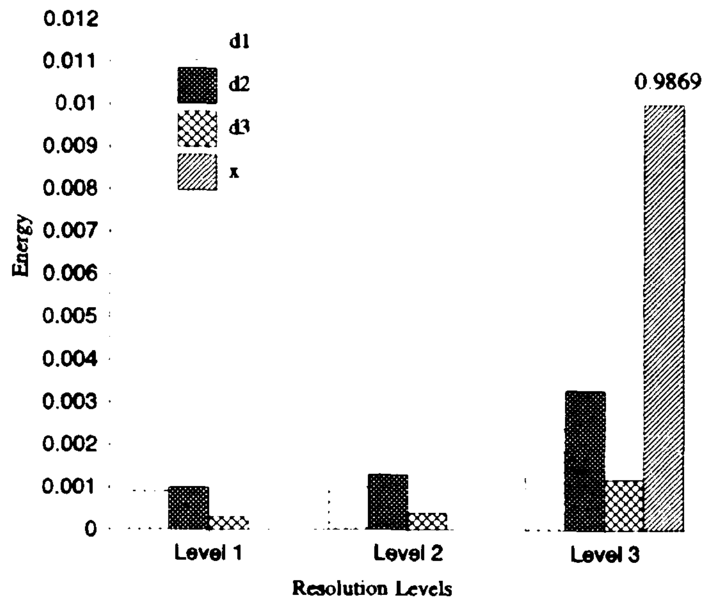

Figure 5.2 Energy Distribution of Wavelet Coefficients Normalized to the Original Signal at Resolution Level $m=0$.

bits among d^i_m by using the bit allocation algorithm of Eq. 4.2. The wavelet coefficients d^i_1, and d^3_2 are discarded due to their low energy values. Keeping the average number of bits at $B=1$ bit per vector, we reserve 11 bits for coding of x_3 and allocate the remaining bits to the rest of the wavelet coefficients. Table 5.3 shows the bit allocations for all wavelet coefficients.

Table 5.3 BIT ASSIGNMENTS TO WAVELET COEFFICIENTS

	Wavelet Coefficients									
	d^1_1	d^2_1	d^3_1	d^1_2	d^2_2	d^3_2	d^1_3	d^2_3	d^3_3	x_3
Codebook Size	0	0	0	3	3	0	8	8	4	11

2. Vector Quantization of Wavelet Coefficients

Given the number of bits for each set of wavelet coefficients, we construct the codebooks by using the GL algorithm. In order to compensate for the decrease in the number of coefficients at lower resolution levels, we use the same approach introduced in Section C.2 of Chapter IV. Consider the signal at resolution level *m* which has roughly one quarter of the samples of that at resolution level *m-1*. First, we decompose the level *m-1* signal into wavelet coefficients. The resulting four wavelet coefficient sets at level *m* are retained. Now, we shift the level *m-1* data by one sample horizontally and repeat the DWT decomposition. The resulting wavelet coefficients are combined with those already obtained at level *m* to extend the training set. This same procedure is repeated using vertical and diagonal shifts. The resulting data set at level *m* has roughly the same size as that at level *m-1*.

After constructing the codebooks using the training data obtained as above, we encode the wavelet coefficients using the corresponding codebooks. In the reconstruction procedure, we substitute zeros for the discarded wavelet coefficients.

C. SIMULATION RESULTS

Simulations were carried out by using two images:

- A training image of 512-by-512 pixels (Fig. 5.3) constructed from pieces of four different images with a resolution of 8 bits per pixel (bpp).
- A test image (Lenna) of 512-by-512 pixels shown in Fig. 5.4.

The test image was not contained in the training data.

The coding performance is measured by using the peak signal-to-noise ratio (PSNR) defined as the ratio of square of the peak input amplitude to the mean square error, D, [Ref. 8]:

$$PSNR = 10\log_{10}(\frac{255^2}{D}) \quad dB \tag{5.1}$$

where:

$$D = d(x,y) = \frac{1}{N^2}\sum_{k=1}^{N}\sum_{l=1}^{N}(x_{kl}-y_{kl})^2 .$$

and N is the number of pixels along one side of the (square) image.

Figure 5.5 shows the performance of WTC in terms of SNR vs. bit-rate, in bits per pixel. We changed the number of pixels in each vector to obtain different bit rates while keeping the number of bits for each quantizer fixed as in Table 5.3. The test image was also coded by DVQ for comparison. The dotted line in Fig. 5.5 represents the performance of DVQ.

Figure 5.6 shows the WTC-coded test image at 0.25 bpp with an SNR of 27.37 dB while Figure 5.7 shows the same image coded by DVQ at 0.25 bpp with an SNR of 20.7 dB. The performance of WTC is very good for a compression rate of 32:1; however, the picture quality is effected by blurred and jagged diagonal edges. This is caused by the smoothing effect of the DWT, which produces a positive effect in speech coding. Using vectors of small size helps to preserve the edges; Figure 5.8 shows the test image coded by WTC with a vector size of 2 pixels (0.5 bpp) and an SNR of 29.5 dB. The quality of the coded image is nearly as good as the original.

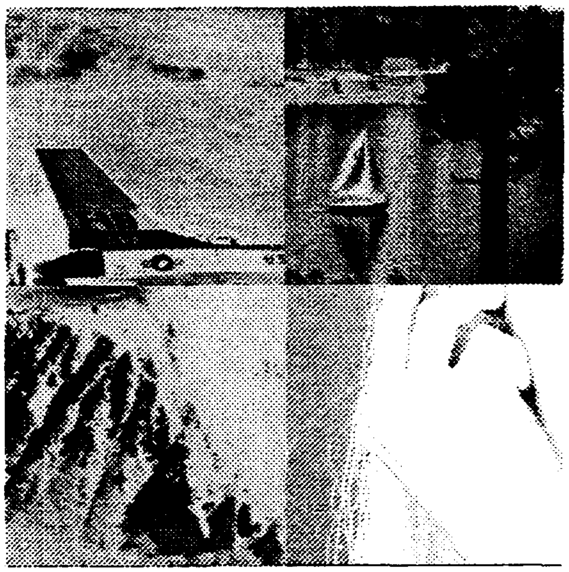

Figure 5.3 Training Image of 512x512.

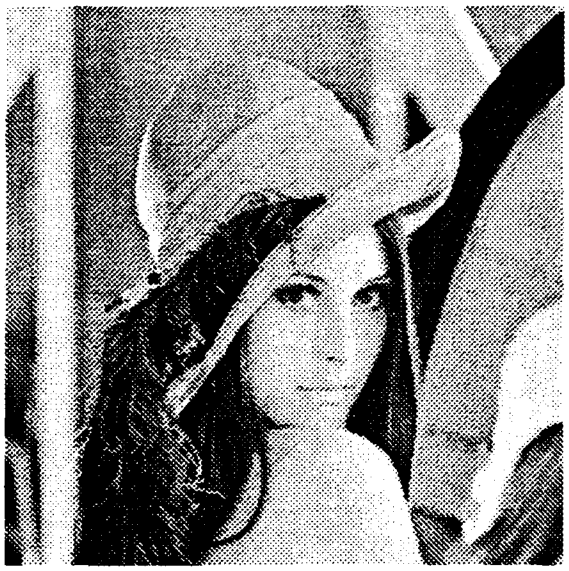

Figure 5.4 Test Image (Lenna) of 512x512.

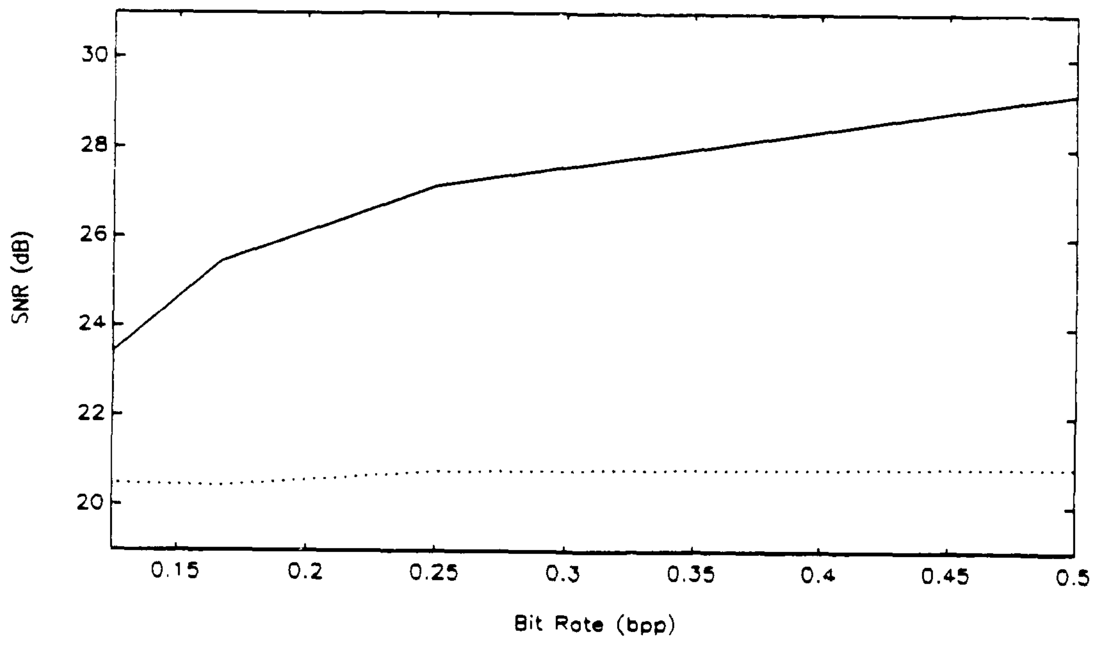

Figure 5.5 SNR vs. Bit Rate: Image Data. The solid line represents WTC while the dotted line represents DVQ.

Table 5.4 illustrates the contribution of each wavelet coefficient set of Figure 5.8 to the overall distortion with the bit rates normalized by W_m (see Eq. 4.2 and 4.4). It is interesting that the contributions of various wavelet coefficients to the overall distortion are by no means equal. It ranges between MSE = 8.24 for d^3_1 and MSE = 172.52 for x_3. Notice the high distortion in x_3 despite the high number of bits per pixel used in the allocation. This underscores the importance of allocating a large number of bits for the codebook of x_3.

Table 5.4 BIT ALLOCATIONS AND MSE CONTRIBUTIONS OF WAVELET COEFFICIENTS OF THE IMAGE IN FIG. 5.8

	d^1_1	d^2_1	d^3_1	d^1_2	d^2_2
Bit Rate (bpp)	0	0	0	0.1013	0.1013
MSE	125.76	131.07	8.24	147.21	171.41
	d^3_2	d^1_3	d^2_3	d^3_3	x_3
Bit Rate (bpp)	0	0.0748	0.0748	0.0374	0.1029
MSE	73.10	67.08	97.38	105.72	172.52

We also coded the test image by using different vector sizes. Since most of the energy was contained in x_3, we used a vector dimension of 2 pixels for x_3 and 8 pixels in each vector for the *d* coefficients. The codebook sizes were kept the same as in Table 5.3. The resulting bit rate was 0.2 bpp with an SNR of 27.15 dB. The reconstructed image is shown in Fig. 5.9. A comparison of Fig. 5.6 and Fig. 5.9 shows that proper use of different vector sizes can result in lower bit rates with better visual quality. We say "better visual quality" instead of "better SNR" because the quality of the image in Fig. 5.9 looks superior to the one in Fig.5.6 although the SNRs are quite close.

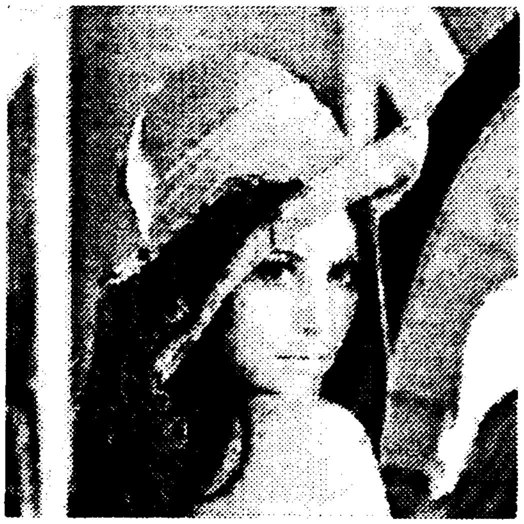

Figure 5.6 Lenna Coded by WTC at 0.25 bpp, SNR = 27.37 dB.

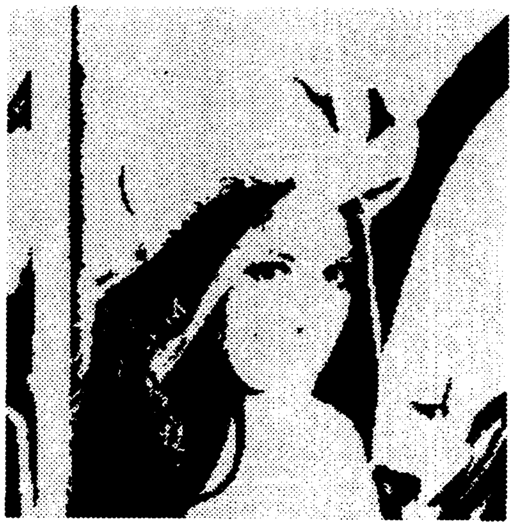

Figure 5.7 Lenna Coded by DVQ at 0.25 bpp, SNR = 20.7 dB.

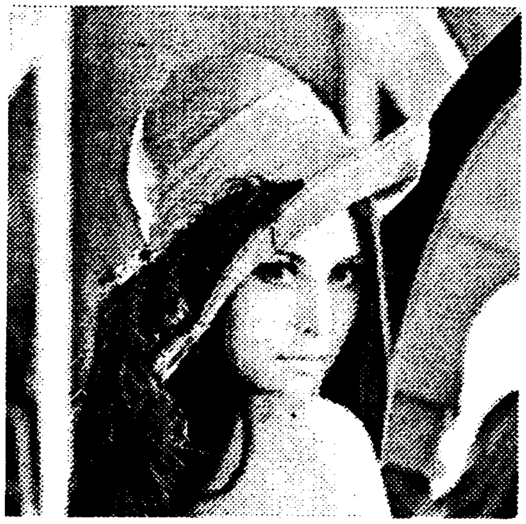

Figure 5.8 Lenna Coded by WTC at 0.5 bpp, SNR = 29.5 dB.

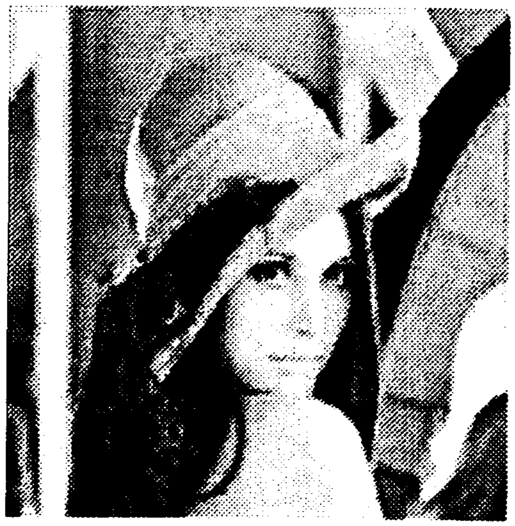

Figure 5.9 Lenna Coded by WTC at 0.2 bpp by Using Different Vector Sizes, SNR = 27.42 dB.

For different simulations, the improvement in performance of WTC over DVQ in terms of SNR ranged from 17 to 50 percent corresponding to bit rates of between 0.16 and 0.5 bpp. The use of different vector sizes for different wavelet coefficients provided better image quality compared to the case of uniform vector size for all wavelet coefficients.

VI. CONCLUSIONS

In this thesis, we combined algorithms for discrete wavelet transforms and vector quantization to develop coding schemes for speech and image data. We studied wavelets based on both orthogonal and biorthogonal bases; we used biorthogonal wavelet functions for image coding since filters with linear phase are desired in image coding.

The algorithm developed for the discrete wavelet transform (DWT) has a pyramidal decomposition architecture and is based on the convolution of data in each pyramid level with wavelet and scaling filters. Unlike those in traditional Fourier theory, the basis functions in the wavelet approach are formed by dilating and translating a single prototype function. Given the coefficients of basis filters with infinite precision, the DWT reconstructs the original data without any error.

Vector quantization was shown here to yield better performance when combined with the DWT. The results as measured in SNR show that the performance has improved by 17% to 84% over direct vector quantization (DVQ) depending on the number of bits per sample and the type of data. In speech coding, we found that the subjective quality of the coded speech was still superior to that using DVQ, even though the improvement in SNR obtained by using the DWT was fairly small. This shows that SNR may not always be a good measure of performance. We also tested the same algorithm on transient signals. The performance of WTC in this case well exceeded that of DVQ.

In image coding, we found that WTC outperforms DVQ for bit-rates around 0.5 bit per pixel (bpp). At bit-rates lower than 0.5 bpp, WTC suffered from blurred and jagged edges although the performance was still better than DVQ. This problem also existed for relatively high bit-rates, but it was not as predominant as it was at low bit-rates.

The major coding gains are due to encoding vectors of samples (vector quantization) and the use of different codebooks for wavelet coefficients at different levels. Although using different codebooks increases the complexity and the computational cost, this research has shown that the VQ-DWT combination improves the quality of the reconstructed data over that of DVQ for a given bit rate (bps or bpp).

APPENDIX A: PROGRAM DETAILS

This appendix contains the program listings for each of the MATLAB algorithms in the thesis. The codes are categorized into three classifications, 1-D DWT, 2-D DWT and VQ.

A. ONE-DIMENSIONAL DWT ROUTINES

```
function lowest=dwt1(c0,qqq,LOW,file,qq)
% DWT1     One-dimensional discrete wavelet transform
%     DWT1(...) computes the wavelet coefficients of vector data C0
%     by using a wavelet filter. It stores the wavelet coefficients
%     in a file and returns the lowest resolution level that the data
%     is processed. It also stores the filter banks to avoid
%     rebuilding them in reconstruction process.
%            C0 = Input data in vector format
%            QQQ = Wavelet filter type
%            LOW = Lowest desired resolution level
%            FILE = Output file containing the wavelet coefficients
%            QQ = Filter tap

% Alper Erdemir
% June 1993

c0=c0';

% Get the filter coeffients
%-------------------------------------------------------------------
if qqq==1
        h=coiflet(qq)';
        wwavelet=['Coif_',num2str(qq)];
elseif  qqq==2,
        h=daubdata(qq);
        wwavelet=['Daub_',num2str(qq)];
else    h=[.5 .5]';wwavelet='HAAR';
end

% Normalize the filter coefficients
%-------------------------------------------------------------------
h=sqrt(2)*h;

% Intialize some of the constants:
```

```
%--------------------------------------------------------------------
LL=length(c0);
Nh=length(h);
lowest = -ceil(log(LL)/log(2));
if abs(lowest) > abs(LOW) lowest=-abs(LOW); end

% Generate the g coefficients
%--------------------------------------------------------------------
g=flipud(h);
for i=2:2:Nh
  g(i,1) = -g(i,1);
end

WIN=256; % window size for data longer than WIN

% The following code constructs the filter banks for the scaling
% filter and the wavelet filter
%--------------------------------------------------------------------
if LL < WIN
        H=fltbnk(h,2*ceil(LL/2));
        G=fltbnk(g,2*ceil(LL/2));
else
        H=fltbnk(h,WIN);
        G=fltbnk(g,WIN);
end

% Decomposition routine
%--------------------------------------------------------------------
for lvl=-1:-1:lowest
        lvl
   eval(['c',num2str(abs(lvl)),'=dcmp1','(c',...
        num2str(abs(lvl)-1),',H,Nh,WIN);'])
        eval(['d',num2str(abs(lvl)),'=dcmp1','(c',...
        num2str(abs(lvl)-1),',G,Nh,WIN);'])
end

% Store the wavelet coefficients
%--------------------------------------------------------------------
temp=[];
for n=1:abs(lowest)
        eval(['temp=[temp ''d',num2str(n),' c',int2str(n),' ''];'])
end
eval(['save ',file,'_',wwavelet,' c0 H G WIN wwavelet Nh ',temp])

% End of routine
%--------------------------------------------------------------------


function x=fltbnk(filt,L)
% FLTBNK Construction of filter matrices for DWT
%       FLTBNK(FILT,L) constructs the filter matrix using the filter FILT.
%       This routine is also used for 2-D DWT.
%               FILT = Filter
%               L = Length of data to be processed

% Alper ERDEMIR
% June 1993

F=length(filt);
Lc=L+2*F-4;
Ll=(Lc-F)/2+1;

x=zeros(Ll,Lc);
for n=1:Ll
        x(n,:)=[zeros(1,2*(n-1)) filt' zeros(1,Lc-F-2*(n-1))];
end

% End of routine
%------------------------------------------------------------------------



function x=dcmp1(data,F,lF,W)
% DCMP1   One-dimensional DWT decomposition routine.
%       DCMP1(...) conducts the DWT decomposition of the input DATA
%       by using the filter bank F with a window size of W.
%               DATA = Input data
%               F = Wavelet/Scaling filter matrix
%               lF = Wavelet/Scaling filter length
%               W = window length for relatively long data

% Alper ERDEMIR
% June 1993

L=length(data);
LF=lF-2;

% The following four lines checks if the data vector is odd, if it is
% true, then a zero is added to it.
%--------------------------------------------------------------------
if rem(L/2,floor(L/2)) ~= 0
        data=[data;0];
        L=L+1;
end
N=fix(L/W);
% Add LF zeros to the beginning and end of DATA
%--------------------------------------------------------------------
data=[zeros(LF,1);data;zeros(LF,1)];

% Decomposition routine
%--------------------------------------------------- -------------
LLL=length(data);
LL=(LLL-lF)/2+1;
```

```
if N >= 1
	Ll1=(W+2*LF-lF)/2+1;
	x(1:Ll1)=F(1:Ll1,1:W+2*LF)*data(1:W+2*LF);
	Ll2=(W+LF-lF)/2+1;
	if N>=2
		for n=1:N-1
			x(Ll1+(n-1)*Ll2+1:Ll1+n*Ll2)=F(1:Ll2,1:W+LF)...
			*data(LF+n*W+1:(n+1)*W+2*LF);
		end
	end
	REM=rem(L,W);
	if REM ~= 0
	Lrem=(REM+LF-lF)/2+1;
		x(LL-Lrem+1:LL)=F(1:Lrem,1:REM+LF)*data(LLL-REM-LF+1:LLL);
	end
	x=x';
else
	x=F(1:LL,1:LLL)*data;
end

% End of routine
%---------------------------------------------------------------------



function ses=idwt1(file)
% IDWT1    One-dimensional inverse discrete wavelet transform
%       IDWT1(FILE) returns the reconstructed data SES by
%       taking the inverse wavelet transform of wavelet coefficients
%       stored in FILE_DWT.

% Alper ERDEMIR
% June 1993

% Lode the wavelet coefficients
%-----------------------------------------------------------------
eval(['load ',file,'_dwt'])

% Start the reconstruction routine
%-----------------------------------------------------------------
eval(['ses=c',num2str(lowest),';'])
FF=Nh-2;
FFF=floor((FF-1)/2+1);
WIN1=WIN-2*FF;
for lvl=-lowest:1:-1
	L=length(ses);
	WIN2=floor((WIN1+Nh-1)/2);
	N=fix((L-FFF)/WIN2);
	eval(['a=length(d',num2str(-lvl-1),');'])
	if N==0
		Lu=(L-1)*2+Nh;
		eval(['dwork=G(1:L,1:Lu)''*d',num2str(-lvl),';'])
```

```
            ses = H(1:L,1:Lu)'*ses + dwork;
            Lp = length(ses)-Nh + 2;
            ses = ses(Nh-2 + 1:Lp);
            ses = ses(1:a);
      else
            Lu = (WIN2-1)*2 + Nh;
            eval(['dwork = G(1:WIN2,1:Lu)''*d',...
                  num2str(-lvl),'(1:WIN2);'])
            cwk(1:Lu) = H(1:WIN2,1:Lu)'*ses(1:WIN2) + dwork;
            Lp = length(cwk)-FF;
            cwk = cwk(Nh-2 + 1:Lp);
            NN = length(cwk);
            if N > = 2
                  Lu2 = Lu + FF;
                  WIN3 = WIN2 + 2*FFF;
                  for n = 1:N-1
                        eval(['dwork = G(1:WIN3,1:Lu2)''*d',...
                              num2str(-lvl),'((n-1)*WIN2-FFF + WIN2 + 1:n*WIN2 + FFF + WIN2);'])
                        cwk2 = H(1:WIN3,1:Lu2)'*ses(n*...
                              WIN2-FFF + 1:(n + 1)*WIN2 + FFF) + dwork;
                        Lp2 = length(cwk2)-FF;
                        cwk(NN + (n-1)*(Lp2-FF) + 1:NN + n*(Lp2-FF)) = cwk2(FF + 1:Lp2);
                  end
                  NN = length(cwk);
            end
            REM = L-N*WIN2;
            if REM ~ = 0
                  Lrem = (REM-1)*2 + Nh;
                  eval(['dwork = G(1:(FFF + REM),1:Lrem)''*d',...
                        num2str(-lvl),'(L-REM-FFF + 1:L);'])
                  cwk2 = H(1:(REM + FFF),1:Lrem)'*ses(L-REM-FFF + 1:L) + dwork;
                  Lp = length(cwk2);
                  cwk(NN + 1:NN + Lp-FF) = cwk2(FF + 1:Lp);
            end
            ses = cwk(1:a)';
      end
end

% End of routine
%- ---------------------------------------------------------------------
```

B. TWO-DIMENSIONAL DWT ROUTINES

```
function lowest = dwt2(c0,flt,LOW,file,qq)
% DWT2     Two-dimensional discrete wavelet transform
%       DWT2(...) computes the wavelet coefficients of two-dimensional
%       data C0 by using a wavelet filter FLT in both horizantal and vertical
%       directions. It stores the wavelet coefficients in a FILE_DWT and
%       returns the lowest resolution level that the data is processed.
%       Both orthogonal and biorthogonal decompostion can be done
%       by using appropriate filters.
%             C0 = Input data in vector format
%           FLT = Wavelet filter type
%           LOW = Lowest desired resolution level
%           FILE = Output file containing the wavelet coefficients
%           QQ -- Filter tap

% Alper Erdemir
% June 1993


% Construct 2-D filter banks for decomposition
%-----------------------------------------------------------------------
[HH,HV,GH,GV,Nh,Nv]=filts2(c0,flt,qq,1);

% Determine of the lowest possible resolution level
%-----------------------------------------------------------------------

[L1 L2]=size(c0);
rsl=min(ceil(log(L1)/log(2)),ceil(log(L2)/log(2)));
lowest=abs(rsl);
if lowest > LOW
        lowest=LOW;
end

% Decomposition routine
%-----------------------------------------------------------------------
for lvl=1:lowest
        eval(['c',num2str(lvl),'=dcmp2(c',...
        num2str(lvl-1),',HH,HV,Nh,Nv);'])
        eval(['d1',num2str(lvl),'=dcmp2(c',...
        num2str(lvl-1),',HH,GV,Nh,Nv);'])
        eval(['d2',num2str(lvl),'=dcmp2(c',...
        num2str(lvl-1),',GH,HV,Nh,Nv);'])
        eval(['d3',num2str(lvl),'=dcmp2(c',...
        num2str(lvl-1),',GH,GV,Nh,Nv);'])
end

% Store the wavelet coefficients
%-----------------------------------------------------------------------
temp=[];
for n=1:lowest
        for m=1:3
                eval(['temp=[temp ''d',int2str(m),int2str(n),' ''];'])
        end
        eval(['temp=[temp ''c',int2str(n),' ''];'])
end
eval(['save ',file,'_dwt flt qq ',temp])

% End of routine
%-----------------------------------------------------------------------
```

```
function [HH,HV,GH,GV,Nh,Nv]=filts2(c0,flt,qq,dir)
% FILTS2 construction of 2-D filter banks
%       FILTS2 returns the 2-D filter matrices using horizantal and
%       vertical wavelet and scaling filters. It also returns the filter
%       lengths.The available filter types are Daubechies 1- to 10-tap
%       filters, Haar filter, and biorthogonal 1- to 5-tap filters.
%               C0 = Input data in matrix format
%               FLT = filter type (for both directions)
%               QQ = Filter tap
%               DIR = Indicator for decomposition and reconstruction procedure
%                     (used for biorthogonal filters)

% Alper ERDEMIR
% June 1993

% Get the filter coefficient
%------------------------------------------------------------------
if flt == 1
        Hh=daubdata(qq);
elseif flt == 2
        Hh=[.5 .5]';
elseif flt == 3
        [Hh,Hh2]=biort(qq);
        Hh=[0;Hh];
        Hh2=[0;Hh2];
        Hv2=Hh2;
end
Hv=Hh;          % Let vertical filter equal to horizantal one

Nh=length(Hh);
Nv=length(Hv);

% Generate the g coefficients
%------------------------------------------------------------------
if flt <= 2
        Gh=flipud(Hh);
        for i=2:2:Nh
                Gh(i,1)=-Gh(i,1);
        end

        Gv=flipud(Hv);
        for i=2:2:Nv
                Gv(i,1)=-Gv(i,1);
        end
else
        if dir == 1
                Gh=flipud(Hh2);
                for i=2:2:Nh
                        Gh(i,1) = -Gh(i,1);
                end
                Gv=flipud(Hv2);
                for i=2:2:Nv
                Gv(i,1) = -Gv(i,1);
            end
        else
            Gh=flipud(Hh);
            for i=2:2:Nh
                Gh(i,1) = -Gh(i,1);
            end
            Gv=flipud(Hv);
            for i=2:2:Nv
                Gv(i,1) = -Gv(i,1);
            end
        Hh=Hh2;Hv=Hv2;
        end
end

[drow0,dcol0]=size(c0);

% Construct the filter banks
%-------------------------------------------------------------------
HH=sqrt(2)*fltbnk(Hh,2*ceil(dcol0/2));
HV=sqrt(2)*fltbnk(Hv,2*ceil(drow0/2));
GH=sqrt(2)*fltbnk(Gh,2*ceil(dcol0/2));
GV=sqrt(2)*fltbnk(Gv,2*ceil(drow0/2));

% End of routine
%-----------------------------------------------------------------------



function x=dcmp2(data,Fh,Fv,lFh,lFv)
% DCMP2  Two-dimensional DWT decomposition Routine
%       DCMP2(...) conducts the DWT decomposition of the input DATA
%       by using horizantal and vertical filter banks, FH and FV,
%               DATA = Input data
%               FH = Horizantal wavelet/scaling filter matrix
%               FV = Vertical wavelet/scaling filter matrix
%               lF = Horizantal wavelet/scaling filter length
%               lF = Vertical wavelet/scaling filter length

% Alper ERDEMIR
% June 1993

[Lv Lh]=size(data);

% If the horizantal length of the data is odd, add a column of zeros
%-------------------------------------------------------------------
if rem(Lh/2,floor(Lh/2)) ~= 0
        data=[data zeros(Lv,1)];
        Lh=Lh+1;
end

% Add lFh-2 columns to left and rigth of the data
```

```
%-------------------------------------------------------------------
data=[zeros(Lv,lFh-2) data zeros(Lv,lFh-2)];

vlen=Lh+2*(lFh-2);       % Width of the Hor. Filt. Matrix
hlen=(vlen-lFh)/2+1;     % Heigth of the Hor. Filt. Matrix

% Horizantal decomposition routine
%-------------------------------------------------------------------
xh=(Fh(1:hlen,1:vlen)*data')';

% If the horizantal length of the data is odd, add a column of zeros
%-------------------------------------------------------------------
if rem(Lv/2,floor(Lv/2)) ~= 0
        xh=[xh;zeros(1,hlen)];
        Lv=Lv+1;
end

% Add lFh-2 columns to left and rigth of the data
%-------------------------------------------------------------------
xh=[zeros(lFv-2,hlen);xh;zeros(lFv-2,hlen)];

vlen2=Lv+2*(lFv-2);          % Width of the Vert. Filt. Matrix
hlen2=(vlen2-lFv)/2+1;       % Heigth of the Vert. Filt. Matrix

% Vertical decomposition routine
%-------------------------------------------------------------------
x=Fv(1:hlen2,1:vlen2)*xh;

% End of routine
%-------------------------------------------------------------------



%function im=idwt2(file)
% IDWT2    Two-dimensional inverse discrete wavelet transform
%       IDWT2(FILE) returns the reconstructed image IM by
%       taking the inverse wavelet transform of wavelet coefficients
%       stored in FILE_DWT.

% Alper ERDEMIR
% June 1993

% Load the wavelet coefficients and construct the filter matrices
%-----------------------------------------------------------------
%eval(['c0=load2(''',file,''');'])
%eval(['load ',file,'_dwt'])
if ~exist('HH')
        [HH,HV,GH,GV,Nh,Nv]=filts2(c0,flt,qq,2);
end

% Reconstruction routine
%-----------------------------------------------------------------
```

```
eval(['im=c',num2str(lowest),';'])
for lvl=lowest:-1:1
        cwrk=rctver2(im,HV,Nv);
        eval(['dwrk1=rctver2(d1',num2str(lvl),',GV,Nv);'])
        datv1=cwrk+dwrk1;
        eval(['dwrk2=rctver2(d2',num2str(lvl),',HV,Nv);'])
        eval(['dwrk3=rctver2(d3',num2str(lvl),',GV,Nv);'])
        datv2=dwrk2+dwrk3;
        dath1=rcthor2(datv1,HH,Nh);
        dath2=rcthor2(datv2,GH,Nh);
        im=dath1+dath2;
        [Lv Lh]=size(im);
        im=im(Nv-2+1:Lv,Nh-2+1:Lh);
        if lvl~=1
                eval(['[a1 a2]=size(d1',num2str(lvl-1),');'])
        else [a1 a2]=size(c0);
        end
        im=im(1:a1,1:a2);
end

% End of routine
%---------------------------------------------------------------



function x=rcthor2(data,Fh,lFh)
% RCTHOR2 conducts the 2-D horizantal reconstruction of
%         wavelet and approximation coefficients.
%                 DATA = Input data
%                 FH = Horizantal wavelet/scaling filter matrix
%                 LFH = Horizantal wavelet/scaling filter length

% Alper ERDEMIR
% June 1993

[Lv Lh]=size(data);
vlen=(Lh-1)*2+lFh;

% do the horizantal decomposition
%---------------------------------------------------------------
x=(Fh(1:Lh,1:vlen)'*data')';

% End of routine
%---------------------------------------------------------------



function x=rctver2(data,Fv,lFv)
% RCTVER2 conducts the 2-D vertical reconstruction of
%         detail and approximation coefficients.
%                 DATA = Input data
%                 FV = Vertical wavelet/scaling filter matrix
%              LFH = Vertical wavelet/scaling filter length

% Alper ERDEMIR
% June 1993

[Lv Lh]=size(data);
vlen=(Lv-1)*2+lFv;

% do the vertical decomposition
%----------------------------------------------------------------
x=Fv(1:Lv,1:vlen)'*data;

% End of routine
%----------------------------------------------------------------
```

C. VQ ROUTINES

```
function w=initcb(x,B)

% INITCB Initialization of the codebook for VQ
%      INITCB(X,B) initializes the codebook randomly selecting data
%      vectors from the subject data set X.
%              X = Data set
%              B = Total number of bits

% Alper Erdemir
% June 1993

N=2^B;  % Find the codebook size

% Randomly select the codewords from X
%----------------------------------------------------------------
rand('uniform')
Nx=max(size(x));
for i=1:N
   w(:,i)=x(:,ceil(Nx*rand(1)));
end

% End of routine
%----------------------------------------------------------------



function x=dat2vec(y,M)

%  DAT2VEC Conversion of data format to be used in VQ
%      DAT2VEC(Y,M) converts the input data into the VQ format
%              Y = Input data of Nx1 (for 1-D) or NxN (for 2-D) where N is
%                 data length
%              M = vector/block size

% Alper ERDEMIR
% June 1993

N = size(y);
n1 = M(1);

% if input data is 1-D
%----------------------------------------------------------------
if N(1) == 1 | N(2) == 1
        in_hei = length(y);
        n1a = ceil(in_hei/n1);
        y = y(:);
        x = [y;zeros(n1*n1a-in_hei,1)];
        x = reshape(x,n1,n1a);

% else, input is 2-D
%----------------------------------------------------------------
else
        n2 = M(2);
        n1a = ceil(N(1)/n1);
        n2a = ceil(N(2)/n2);
        x = zeros(n1*n2,n1a*n2a);
        yy = zeros(n1a*n1,n2a*n2);
        yy(1:N(1),1:N(2)) = y;
        for i = 1:n1a
                i1 = (i-1)*n1 + 1;
                for j = 1:n2a
                        k = (i-1)*n2a + j;
                        j1 = (j-1)*n2 + 1;
                        z = yy(i1:i1 + n1-1,j1:j1 + n2-1);
                        x(:,k) = z(:);
                end
        end
end

% End of routine
%----------------------------------------------------------------


function x = vec2dat(y,N,BLK)
% VEC2DAT conversion of the data from VQ format to original format
%       VEC2DAT(Y,N,BLK) converts the data Y in VQ format to original
%       format of size Nx1 (for 1-D) or size NxN (for 2-D).
%               Y    = Subject data in vector format
%               N    = Size of original data
%               BLK  = Vector/block size

% Alper Erdemir
% June 1993

% if data is 1-D
%----------------------------------------------------------------
```

```
if nargin == 2
        y=y(:);
        x=y(1:N);

% else it is 2-D
%----------------------------------------------------------------
else
        n1a=ceil(N(1)/BLK(1));
        n2a=ceil(N(2)/BLK(2));
        x=zeros(BLK(1)*N(1),BLK(2)*N(2));
        for i=1:n1a
                i1=(i-1)*BLK(1)+1;
                for j=1:n2a
                        k=(i-1)*n2a+j;
                        j1=(j-1)*BLK(2)+1;
                        z=zeros(BLK(1),BLK(2));
                        for l=1:BLK(2)
                                m=(l-1)*BLK(1)+1;
                                z(:,l)=y(m:m+BLK(1)-1,k);
                        end
                        x(i1:i1+BLK(1)-1,j1:j1+BLK(2)-1)=z;
                end
        end
        x=x(1:N(1),1:N(2));
end

% End of routine
%----------------------------------------------------------------



function [w,m]=lbg(tr,bit,BLK)
% LBG conducts the generalized Lloyd's algorithm
%       LBG(TR,BIT,BLK) returns the trained codebook for VQ. The code first
%       initializes the codebook, then trains it by using Lloyd's iteration.
%       The training stops when the criteria equals to 0.01.
%               TR = Training data
%               BIT = Total number of bits to costruct the codebook
%               BLK = Vector/block size

% Alper Erdemir
% June 1993

x=dat2vec(tr,BLK);

% Initialize the codebook
%----------------------------------------------------------------
w=initcb(x,bit);
m=[];
N=0;

% Do the Lloyd's iteration until convergence
```

```
%-----------------------------------------------------------------
for n=1:2
        w=lloyd(x,w);
        m=mse(x,w,m);
save inf m w
        N=N+1;
end
while 1
        w=lloyd(x,w);
        m=mse(x,w,m);
save inf m w
        N=N+1;
        if abs((m(N-1)-m(N))/m(N-1)) <= 0.01
                break;
        end
end

% End of routine
%-----------------------------------------------------------------



function wup=lloyd(x,w)
% LLOYD Lloyd's iteration for VQ
%       LLOYD(X,W) performs the Lloyd iteration for codebook improvement
%       to the given codebook W by using the training data set X.
%               X = Training data
%               W = Previous codebook

% Alper ERDEMIR
% June 1993

L=size(x);
N=size(w);
wup=zeros(N(1),N(2));
u=zeros(N(2),1);
y=ones(1,N(2));

% Partition the training set into clusters by using NNC
%-------------------------------------------------------
for k=1:L(2)
        d=sum((x(:,k)*y-w).^2);
        [na,iw]=min(d);
        wup(:,iw)=wup(:,iw)+x(:,k);
        u(iw)=u(iw)+1;
end

% Take care of the empty clusters ( <=3 training vectors)
% by splitting the centroid of the clusters with
% the highest numbers of training vectors
%-------------------------------------------------------
i=find(u==0);
```

```
u(i)=ones(size(i));
wup=wup./(u*ones(1,L(1)))';
i=find(u<4);
[Y I]=sort(u);
I=flipud(I);
for n=1:length(i)
        eps=norm(wup(:,I(n)))*.001;
        wup(:,i(n))=wup(:,I(n))+eps;
        wup(:,I(n))=wup(:,I(n))-eps;
        u(i(n))=u(I(n));
end

% End of routine
%----------------------------------------------------------------



function m=mse(x,w,m)

% MSE Computation of mean-square error of a codebook
%       MSE(X,W,M) computes the mean-square error of the given codebook
%       W, and returns it  by appending the current value to an existing
%       vector M, which has the MSE record for each iteration of
%       the codebook
%               X = Training data
%               W = Current codebook
%               M = Vector containing the MSE record

% Alper Erdemir
% June 1993

N=size(w);
Nx=max(size(x));
mse1=0;
y=ones(1,N(2));

for k=1:Nx
        d=sum((x(:,k)*y-w).^2);
        mse1=mse1+min(d);
end
mse1=mse1/(Nx*N(1));
m=[m,mse1];

% End of routine
%----------------------------------------------------------------



function [z,mse]=code(x,w)
% CODE coding of data set X using codebook W
%       CODE(X,W) returns the encoded data, Z, and the mean-square error
%       of coding.
%            X = Data set to be coded
%            W = Codebook

% Alper ERDEMIR
% June 1993

N=size(w);
Nx=max(size(x));
mse=0;
y=ones(1,N(2));
z=zeros(N(1),Nx);

% Find the closest code vector to each input vector
%----------------------------------------------------------------
for k=1:Nx
   d=sum((x(:,k)*y-w).^2);
   [md,iw]=min(d);
   z(:,k)=w(:,iw);
   mse=mse+md;
end

% Find the mean-square error
%----------------------------------------------------------------
mse=mse/(Nx*N(1));

% End of routine
%----------------------------------------------------------------
```

LIST OF REFERENCES

1. Olivier Rioul and Martin Vetterli, "Wavelets and Signal Processing," *IEEE Signal Processing Magazine*, pp. 14-28, October 1991.

2. Ingrid Daubechies, *Ten Lectures on Wavelets*, Society of Industrial and Applied Mathematics, Philadelphia, 1992.

3. Stephane G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intellegence*, Vol. 11, No. 7, pp. 674-693, 1989.

4. Martin Vetterli and Cormac Herley, "Wavelets and Filter Banks: Theory and Design," *IEEE Transactions on Signal Processing*, Vol. 40, No. 9, 1992.

5. Ali N. Akansu and Richard A. Haddad, *Multiresolution Signal Decomposition*, Academic Press, San Diego, 1992.

6. Robert M. Gray, "Vector Quantization," *IEEE ASSP Magazine*, Vol.1, pp.4-29, April 1984.

7. John Makhoul and others, "Vector Quantization in Speech Coding," *Proceedings of the IEEE*, Vol. 73, No.11, pp. 1551-1558, November 1985.

8. Allen Gersho and Robert M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Boston, 1992.

9. Y. Linde and Others, "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications*, Vol. COM-28, pp. 84-95, January 1980.

10. M. Antonini, and others, "Image Coding Using Vector Quantization in the Wavelet Transform Domain," *IEEE International Conference on Acoustics, Speech, and Signal Processing, Albuquerque*, pp. 2297-2300, April 1990.

11. M. Antonini, and Others, "Image Coding Using Lattice Vector Quantization of Wavelet Coefficients," *IEEE International Conference on Acoustics, Speech, and Signal Processing, Toronto*, pp. 2273-2276, May 1991.

12. Ronald A. DeVore, Bjorn Jawerth, and Bradley J. Lucier, "Image Compression Through Wavelet Transform Coding," *IEEE Transactions on Information Theory*, Vol. 38, No. 2, pp. 719-746, March 1992.

13. Olivier Rioul and Pierre Duhamel, "Fast Algorithms for Discrete and Continuous Wavelet Transforms," *IEEE Transactions on Information Theory*, Vol. 38, No. 2, pp. 569-586, March 1992.

14. Ingrid Daubechies, "Orthonormal Bases of Compactly Supported Wavelets," *Communications on Pure and Applied Mathematics*, Vol. XLI, pp. 909-996, 1988.

15. C. E. Shannon, "Coding Theorems for a Discrete Source with a Fidelity Criterion," *IRE National Convention Record*, Part 4, pp.142-163, 1959.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria VA 22304-6145	2
2.	Library, Code 052 Naval Postgraduate School Monterey CA 93943-5002	2
3.	Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
4.	Professor Murali Tummala, Code EC/Tu Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	2
5.	Professor Charles W. Therrien, Code EC/Ti Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
6.	Samuel J. Frazier, Code SY84 Aircraft Division Naval Air Warfare Center Patuxent, MD 20670-5000	1
7.	Deniz Kuvvetleri Komutanligi Personel Daire Baskanligi Bakanliklar, Ankara/TURKEY	2
8.	Golcuk Tersanesi Komutanligi Golcuk, Kocaeli/TURKEY	1

9.	Deniz Harp Okulu Komutanligi 81704 Tuzla, Istanbul/TURKEY	1
10.	Taskizak Tersanesi Komutanligi Kasimpasa, Istanbul/TURKEY	1
11.	LTJG Alper Erdemir Gunesli Sok. No: 4 Daire: 7 81090 Erenkoy, Istanbul/TURKEY	1